Universidad Autónoma de Madrid
Escuela politécnica superior

Doble Grado en Ingeniería Informática y Matemáticas

Trabajo Fin de Grado

DESARROLLO DE UNA GUI PARA EL ANÁLISIS DE DATOS DE SECUENCIACIÓN GENÓMICA

Autora: Susana Hernández Ballesteros
Tutora: Irene Rodríguez Luján
Ponente: José Ramón Dorronsoro Ibero

Enero de 2016

DESARROLLO DE UNA GUI PARA EL ANÁLISIS DE DATOS DE SECUENCIACIÓN GENÓMICA

Autora: Susana Hernández Ballesteros
Tutora: Irene Rodríguez Luján
Ponente: José Ramón Dorronsoro Ibero

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Enero de 2016

Agradecimientos

Me gustaría agradecer en primer lugar a mi tutora, Irene Rodríguez Luján, la oportunidad que me ha brindado de realizar este proyecto. Ha sido gracias a su ayuda y apoyo, a sus magníficas directrices y a sus desvelos que este proyecto haya llegado a buen puerto.

Así mismo, quería agradecer a todas las personas que se han implicado en la realización de este proyecto, aportando ideas, críticas y mejoras, para que el proyecto se completase de la mejor forma posible y fuese útil a todas aquellas personas a las que está orientado. También le quiero agradecer a Luis Rodríguez Luján su aportación sobre Shiny.

Además quiero agradecerles a todos los compañeros que me me han acompañado a lo largo de la carrera, en los momentos buenos y malos, en los momentos de ocio y de estudio, de quienes me llevo un bonito recuerdo y, en algunos de los casos, una sincera amistad.

Querría hacer especial mención a aquellos más cercanos, Julia, Ana, Deby, Carmelo, Alberto, Álvaro y el chico con el que hacía las practicas de Análisis de Algoritmos, Javi, (¿o eran las de SO?), con quienes tengo el placer de decir que he compartido 5 de los mejores y más importantes años de mi vida tanto en desarrollo intelectual como personal. También me acuerdo en todo momento de mis queridísimas amigas de la infancia, que perduran en su amistad y a las que adoro, Rebeca, Diana y Ángela que siempre han confiado en mi capacidad para conseguir lo que me proponía más que yo misma.

Y por último no me puedo olvidar de los grandes pilares que han soportado esta dura carrera de fondo, mis padres, José María y Susana, y mi hermana María, quienes desde el primer día de universidad me han dado los ánimos, el apoyo y el cariño necesario para poder superar cada uno de los obstáculos con los que me he encontrado. Sin ellos esto no habría sido posible. Gracias de todo corazón por todo.

Resumen

RNA-seq es una tecnología perteneciente al grupo de secuenciaciones de nueva generación (Next Generation Sequencing, NGS), que proporciona información sobre el ARN y, en particular, sobre la expresión de genes presentes en dicho ARN. Se trata de una tecnología ampliamente utilizada para realizar el análisis de la expresión diferencial de genes. Sin embargo, la principal dificultad de este tipo de análisis reside en la falta de un *workflow* unificado, debido a la gran cantidad de herramientas que existen y que su uso no es sencillo para los usuarios no expertos en informática. El uso de estas herramientas a día de hoy se realiza mediante la línea de comandos UNIX y es necesario conocer los formatos de de los ficheros que se utilizan en cada paso del análisis RNA-seq.

El objetivo de este Trabajo de Fin de Grado es crear un *workflow* que integre algunas de las herramientas empleadas en cada paso del análisis RNA-seq y permitir su utilización mediante una Interfaz Gráfica de Usuario (GUI) que facilite realizar el análisis de expresión diferencial de genes. Esto se ha conseguido en dos partes. Por una parte, se ha realizado un estudio exhaustivo de las herramientas disponibles en el estado del arte para determinar cuales son las más utilizadas por los usuario finales. Las herramientas seleccionadas son: Bowtie, Samtools, HTSeq y DESeq. Además, se ha diseñado y desarrollado contador de expresión génica que trata de paliar las deficiencias de algunas de las herramientas más populares. Por otra parte, se ha desarrollado una aplicación web que permite realizar el análisis de expresión diferencial de varias condiciones experimentales simultáneamente, integra todas las herramientas seleccionadas, permite determinar las opciones de cada una de ellas y proporciona al usuario una visualización gráfica de los resultados del análisis de expresión diferencial.

La Interfaz Gráfica de Usuario utiliza las librerías Shiny y Shinydashboard lo que mejora la usabilidad y la experiencia de usuario en comparación a la utilización de las herramientas por la línea de comandos o a través de herramientas de integración demasiado sofisticadas que no llegan a ser útiles para el usuario no experto. La aplicación ha sido diseñada e implementada de acuerdo al patrón de arquitectura Modelo-Vista-Controlador (MVC). La funcionalidad de la vista y el controlador vienen proporcionados por la estructura que sigue la implementación con Shiny y Shinydashboard. El modelo queda definido por las herramientas utilizadas para el análisis y por el formato estándar de los ficheros generados durante el análisis de expresión diferencial de genes.

Finalmente, los requisitos funcionales y no funcionales de la aplicación han sido validados y verificados realizando pruebas con datos RNA-seq simulados y experimentos reales.

Palabras Clave — Interfaz gráfica de usuario, RNA-seq, NGS-secuenciaciones de nueva generación, análisis de expresión diferencial de genes, contador de genes, Shiny, Shinydashboard, Bowtie, HTSeq, DESeq.

Abstract

RNA-seq is a Next-Generation Sequencing (NGS) technology that provides information about RNA and in particular the gene expression present in a sample of RNA. It is a widespread and useful method for the analysis of differential gene expression. However, the large number of tools available and the lack of an unified workflow make the RNA data analysis very difficult to non-experts.

The objective of this Masters Thesis is to create a work flow that integrates some of the most popular tools used in each step of RNA-seq analysis and allow its use through a Graphical User Interface (GUI) that facilitates the gene differential expression analysis. This is achieved by completing two main tasks. First, conducting a study of the state-of-the-art RNA-seq tools in order to determine which tools are the most used by the end user, learn how to use them. Namely, the selected tools are: Bowtie, Samtools, HTSeq and DESeq. A tool for gene expression counting was also implemented and included in the workflow in order to alleviate some disadvantages of the existing algorithms. Second, developing a web application that allows the analysis of differential gene expression of various experimental conditions simultaneously. The GUI integrates all tools selected and allow tuning the configuration parameters of each tool. Additionally, it provides a graphical display with the results obtained through the analysis.

The GUI uses the R Shiny and Shinydashboard libraries, and it improves usability and user experience compared to the alternative of using command-line tools or sophisticated integration tools not suitable for non-experienced users. The application was designed and implemented according to a model-view-controller architectural pattern. The view and controller functionalities are directly provided by Shiny and Shinydashboard libraries. The model consists of the selected tools, and the data model is defined by the standard file formats generated during the analysis of differential gene expression.

Finally, functional and non-functional requisites of the application were validated using RNA-seq data from simulations and real experiments.

Keywords — Graphical User Interface, RNA-seq, Next Generation Sequencing Tools, differential gene expression analysis, gene counting, Shiny, Shinydashboard, Bowtie, HTSeq, DESeq.

Glosario

Bowtie Alineador de lecturas para un gran conjunto de secuencias cortas de ADN a genomas de gran tamaño. 6, 14, 15, 17, 19, 22, 23, 27, 31, 32, 34, 47, 49, 50, 54, 56–58, 77, 93, 96, 98, 99

DESeq Análisis diferencial de la expresión génica basada en la distribución binomial negativa. Paquete de R con funciones para realizar este tipo de análisis. 6, 7, 9, 11, 15, 16, 18, 19, 23, 24, 26, 27, 32, 34, 37, 39, 47, 51, 77, 94, 101

FASTA Formato de fichero informático basado en texto, utilizado para representar secuencias de ácidos nucleicos. 13, 14, 18, 29, 34, 37, 50, 71

FASTQ Formato de fichero informático basado en texto, utilizado para representar secuencias de ácidos nucleicos junto con las correspondientes calidades de dichas secuencias. 13, 14, 18, 34, 37, 50, 71, 72

HTSeq Paquete de Python que proporciona la infraestructura necesaria para procesar los datos de los ensayos de secuenciación de alto rendimiento. 6, 9, 15, 18, 19, 22, 25, 32, 34, 37, 47, 51, 77, 87, 90, 91

microarrays Superficie sólida a la cual se une una colección de fragmentos de ADN. 2

RNA-seq Tecnología de secuenciación de nueva generación para el análisis de expresión diferencial de genes. 2–6, 8, 9, 11, 13, 14, 16, 17, 33, 34, 37, 77

Samtools Conjunto de programas para interactuar con los datos de secuenciación de alto rendimiento. 6, 18, 19, 22, 32, 47, 50, 77, 79

Shiny *Framework* de aplicaciones web para R. 11, 21, 25, 27, 28, 30, 37, 47

Acrónimos

ADN Ácido desoxirribonucleico. 1, 8, 71, 74

ARN Ácido ribonucleico. 1, 2, 5, 71, 74

BAM Binary Alignment/Map. 6, 22, 32, 77

BN Binomial Negativa. 7

BWT Burrows–Wheeler transform. 6

FDR False Discovery Rate. 7, 15, 18, 19, 24, 36, 100

GFF General Feature Format. 6, 13–15, 18–22, 37, 51, 74, 75

GTF General Transfer Format. 6, 74

GUI Graphical User Interface. 4, 11, 30

HTGS High Throughput Genome Sequencing. 6

IDE Integrated Development Environment. 11, 47

MVC Modelo-Vista-Controlador. 17, 25, 37, 38

NGS Next-Generation Sequencing. 2, 6, 78

SAM Sequence Alignment/Map. 6, 13–15, 18–22, 26, 32, 37, 50, 72, 74, 75, 77

TFG Trabajo de Fin de Grado. 22, 34, 83, 87, 93

Índice general

Índice de Figuras

Índice de Tablas

1 Introducción

1.1. Motivación del proyecto

La bioinformática es un campo interdisciplinar que desarrolla metodologías y herramientas que ayudan a gestionar, analizar y entender datos biológicos tales como pueden ser el Ácido desoxirribonucleico (ADN) o el genoma. Esta ciencia integra disciplinas tales como la biología, la matemática, en particular la estadística, y la ingeniería informática [6].

En todos y cada uno de los organismos vivos, el ADN codifica toda la información necesaria para determinar las propiedades y funciones de cada célula. La expresión de genes permite a cada célula reproducir sus instrucciones mediante la activación y desactivación de ciertos grupos de genes. La información codificada en los genes se transforman en moléculas de Ácido ribonucleico (ARN), las cuales pueden ser usadas para el análisis de la expresión de genes. Como se muestra en la figura 1.1, esta transcripción luego se traduce en las diferentes proteínas producidas por el ARN.

Los principales esfuerzos de la investigación en este campo incluyen la secuenciación del genoma y la expresión diferencial de genes. Es por ello que la secuenciación del ADN [7] y el análisis de los datos obtenidos de la misma es un área de gran interés y constituye hoy en día uno de los campos de la biología en pleno desarrollo, el cual sólo es posible gracias a la ayuda de las ciencias de la computación y la matemática. Este trabajo se centrará en el análisis de expresión diferencial de genes.

El análisis de la expresión diferencial se basa en el estudio de muestras de ARN, en formato de *reads* (o lecturas), de un organismo sometido a ciertas condiciones experimentales, las cuales se pretenden alinear con el genoma de referencia para posteriormente cuantificar la cantidad de expresión de cada uno de los genes anotados en dicho genoma. Esta comparación permite establecer relaciones de causalidad entre las muestras analizadas a partir de la comparativa en la expresión de genes entre diferentes condiciones experimentales. Normalmente se realizan comparativas de los resultados obtenidos al contabilizar la expresión de los genes bajo una condición de control frente a su expresión en otras condiciones experimentales.

Hoy en día, una constante en proyectos de bioinformática y biología computacional es el uso

de herramientas informáticas y matemáticas para extraer información útil de datos producidos por técnicas de secuenciación del genoma del alta productividad, también denominadas tecnologías de siguiente generación, (Next Generation Sequencing, NGS)[8]. Estas técnicas han sido ampliamente adoptadas desde 2007 y se basan en la secuenciación masiva en paralelo, logrando reducir significativamente el coste de las secuenciaciones además de proporcionar de forma rápida datos mucho más precisos que tecnologías utilizadas hasta el momento, como microarrays. Las tecnologías NGS se caracterizan por proveer una gran cantidad de lecturas pequeñas y aleatorias del genoma, lo que garantiza una buena cobertura del genoma del organismo a analizar. Existen varias tecnologías englobadas dentro del término NGS, centrándonos en este proyecto en el estudio de la secuenciación y análisis del ARN mediante una técnica llamada RNA-seq [9] dado que está ampliamente extendida y es especialmente apropiada para el análisis de la expresión diferencial de genes.

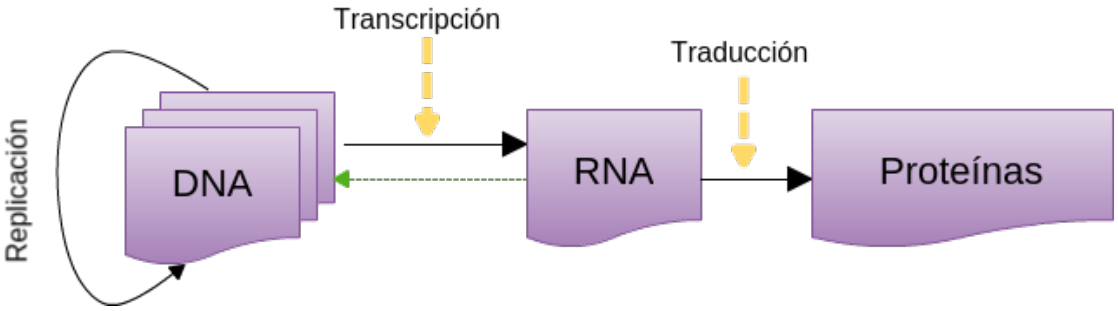


Figura 1.1: Dogma central de la biología molecular

Las tecnologías NGS conllevan algunos desafíos para la bioinformática [10]. Uno de ellos es el manejar grandes cantidades de datos en cada ejecución. También nos encontramos con un problema computacional, ya que el volumen de datos a analizar puede ser demasiado grande para un ordenador. Otro problema que se presenta en este tipo de análisis es la falta de un gran número de réplicas de cada una de las condiciones experimentales debido a su alto coste, lo que representa un reto en el análisis de los datos y la obtención de resultados estadísticamente significativos. Este trabajo se centrará en otro de los grandes problemas que se presenta hoy en día en las herramientas de análisis de datos RNA-seq: su usabilidad e integración. Actualmente no está disponible un software estándar para usuarios finales no expertos en algoritmos y técnicas de análisis de datos RNA-seq y que deseen obtener unas primeras conclusiones de los resultados de sus experimentos utilizando herramientas estándar *de facto*. Además, el análisis diferencial requiere el uso concatenado de diferentes herramientas: filtrado de lecturas, alineación, conteo y análisis de expresión diferencial. Estas herramientas tienen diversos parámetros a ajustar los cuales son determinantes en las conclusiones obtenidas de los experimentos, pero la configuración de estos parámetros requiere un conocimiento en detalle de las opciones de cada programa que forma el *workflow* (*pipeline*) del análisis, las cuales distan mucho de ser estándar. Además, estas herramientas están escritas en diferentes lenguajes de programación y requieren de un cierto nivel de conocimiento informático ya que la mayoría se ejecutan y controlan desde una terminal UNIX. Estas herramientas serán comentadas más en detalle en el Capítulo 2, donde se hará un estudio del estado del arte de este tipo de análisis.

1.2. Objetivos y enfoque

Estos dos problemas, la necesidad de un profundo conocimiento de las herramientas y la falta de un entorno amigable y sencillo para su uso e integración, pueden suponer un grave dilema al usuario a la hora de decidir que software utilizar para el procesamiento y análisis de los datos. Como resultado de estas observaciones, el objetivo principal del proyecto es desarrollar una interfaz gráfica de usuario que integre herramientas para el análisis comúnmente utilizadas y que permita a un usuario sin conocimientos de informática avanzados el poder hacer uso de las mismas. La interfaz gráfica abarcará el proceso de análisis de los datos RNA-seq desde la alineación al cálculo de la expresión diferencial, facilitando un servicio que permita completar todo el proceso desde un único punto ofreciendo además una representación de los resultados obtenidos tras el análisis. En definitiva, facilitar al usuario un servicio sencillo e intuitivo para el procesamiento y análisis de de expresión diferencial de genes.

Para alcanzar este objetivo se han establecido los siguientes objetivos parciales:

- Estudio detallado del estado del arte
  Se realizará un análisis de las herramientas disponibles en cada uno de los pasos del pipeline estándar para el análisis de expresión diferencial. Se estudiarán las características principales de las herramientas más populares sopesando pros y contras, para incluirlas en el producto final.

- Análisis detallado de las herramientas seleccionadas
  Se procederá al estudio de las herramientas y se realizará un aprendizaje de uso de las mismas. El objetivo será conseguir realizar un análisis completo de expresión diferencial a partir de genomas de referencia, lecturas y otros datos proporcionados por el usuario a partir de las herramientas seleccionadas y poder reproducir resultados correctos.

- Diseño e implementación
  Se diseñará e implementará una interfaz gráfica que permita unificar las herramientas estudiadas anteriormente y se implementará un contador de expresión génica. La interfaz gráfica deberá ser capaz de reproducir la funcionalidad propia del proceso de análisis de expresión diferencial con RNA-seq además de resolver las dificultades de usabilidad e integración expuestas en la Sección 1.1.

- Evaluación
  Se procederá a hacer un análisis a partir de un plan de pruebas que permita determinar el correcto funcionamiento de la aplicación desarrollada utilizando tanto datos obtenidos a partir de simulaciones como datos de experimentos RNA-seq reales.

Establecida la motivación del proyecto y el objetivo principal del mismo, el desarrollo de la interfaz de usuario para el análisis de datos de RNA-seq requerirá abordar las siguientes tareas:

- Recopilación de información de las herramientas disponibles y sus propiedades.
- Selección de las herramientas de secuenciación, alineación, recuento, y expresión diferencial de genes.
- Estudio en profundidad de las herramientas seleccionadas.
- Desarrollo e implementación de una herramienta para el conteo de genes que nos permita profundizar el el proceso de análisis.

- Desarrollo de una interfaz gráfica de usuario, GUI, que recoja y ofrezca la funcionalidad de las herramientas seleccionadas.
- Representación gráfica de los resultados obtenidos mediante el procesamiento aplicado a los datos resultantes.
- Análisis de los resultados obtenidos mediante el procesamiento de los datos gracias a la aplicación y validación de los mismos.

1.3. Estructura de la memoria

- **Capítulo 1:** Introducción y motivación del proyecto.
- **Capítulo 2:** Estado del Arte. En este capítulo se explican algunas de las herramientas que se utilizan en este proyecto para el análisis de expresión diferencial usando RNA-seq.
- **Capítulo 3:** Definición del proyecto. Se describe el alcance del proyecto, la metodología seguida durante el ciclo de vida y las herramientas que se han utilizado para llevarlo a cabo.
- **Capítulo 4:** Requisitos. Se realiza una enumeración de los requisitos funcionales y no funcionales de la interfaz gráfica de usuario y de las herramientas desarrolladas.
- **Capítulo 5:** Diseño. En este capítulo se describe el diseño de la aplicación, incluyendo el patrón de la arquitectura, la herramienta para el conteo de expresión diferencial de genes y la interfaz gráfica.
- **Capítulo 6:** Implementación. Se explica cómo se han implementado los diferentes módulos de la aplicación.
- **Capítulo 7:** Pruebas y evaluación. Se detallan las pruebas llevadas a cabo sobre el proyecto que permiten validar y verificar el correcto funcionamiento del mismo.
- **Capítulo 8:** Conclusiones y trabajo futuro. Se exponen las conclusiones a las que se ha llegado tras la finalización del trabajo. Posibles mejoras de la aplicación y líneas de trabajo a seguir tras el fin del Trabajo de Fin de Grado.
- **Referencias, glosario y anexos**.

2

Estado del arte

Antes de presentar nuestro desarrollo vamos a hacer un breve estudio del estado del arte en el campo de análisis de expresión diferencial con RNA-seq. Lo que se pretende con este estudio es formar una idea de las distintas herramientas existentes para el análisis RNA-seq y conocer algunos de los algoritmos y modelos estadísticos que se aplican. La combinación de estas herramientas constituye lo que se denomina *pipeline* (o *workflow*) para el análisis de expresión diferencial y la interfaz gráfica pretende abarcar la funcionalidad de esta *pipeline*.

2.1. Expresión diferencial de genes basada en RNA-seq

RNA-seq es una metodología para el análisis del Ácido ribonucleico (ARN) basado en la nueva generación de secuenciación que permite medir y comparar patrones de expresión de genes. Las características de esta técnica han hecho posible el uso de esta metodología en un amplio rango de estudios. Sin embargo, la rápida evolución de los protocolos y la aparición en paralelo de numerosas herramientas de computación han comprometido la adopción de un *pipeline* estándar para el análisis de expresión diferencial con RNA-seq [11] [12] [13] [14] [2].

En la figura F.2 del apéndice F se pueden observar los pasos que se siguen durante un análisis de expresión diferencial basado en RNA-seq, desde los datos iniciales hasta la obtención de resultados. Los pasos del *pipeline* que se muestran son:

1. Recogida de los datos tales como el genoma de referencia y los *reads* asociados a cada experimento. También se ha de establecer el conjunto de software a utilizar durante el análisis (paso 1).

2. Establecer el conjunto de softwares que se utilizarán para el análisis.

3. Controles de calidad de los datos. En esta etapa se descartan todas aquellas lecturas que no verifiquen un mínimo de calidad establecido por el usuario (paso 2).

4. Obtención de metadatos de los experimentos tales como la condición experimental asociada y la configuración experimental utilizada en la secuenciación (pasos 3 al 6).

5. Alineación de los *reads* frente al genoma de referencia (pasos 7 al 12).
6. Conteo de los genes (paso 13).
7. Análisis de la expresión diferencial normalizando previamente los datos (paso 14).
8. Comprobaciones adicionales sobre la validez de los resultados (paso 15). Este paso es opcional.

Para cada uno de estos pasos se pueden utilizar distintas herramientas existentes en la literatura. A continuación se procede al estudio de las herramientas seleccionadas para algunas de las fases del análisis, ya que en este proyecto no se incluirá el control de calidad de los datos y la recogida de datos y metadatos se realizará a través de la interfaz gráfica.

2.1.1. Alineación: Bowtie

La alineación de las lecturas (pasos 7-12 de la figura F.2) consiste en determinar la posición del genoma de referencia que originó dicha lectura. Una de las principales características comunes de los alineadores de lecturas RNA-seq que utilizan lecturas cortas, típicamente del orden de cientos de bases, mapeadas al genoma de referencia, cuya longitud es en general varios órdenes de magnitud superior a la longitud de las lecturas. Este proceso se encuentra con varios retos, como garantizar la eficiencia a la hora de determinar donde encaja una secuencia (no necesariamente de forma exacta) a la vez que tienen que determinar cuando las divergencias encontradas se deben a errores técnicos producidos por métodos de secuenciación como Illumina [15], o por el contrario se deben efectivamente a variaciones de la muestra.

Desde el comienzo del auge de la tecnología NGS en el año 2007, han aparecido numerosos alineadores tal y como muestra la figura D.1 del apéndice D. La revisión del estado del arte llevada a cabo en [2] señala a Bowtie como uno de los alineadores más populares para datos RNA-seq, siendo el algoritmo con el mayor número de citaciones por año de publicación [16]. Bowtie utiliza Burrows–Wheeler transform (BWT) para crear índices permanentes del genoma. Es capaz de alinear más de 25 millones de lecturas generadas con Illumina en 1 hora de CPU, además puesto que trabajaremos con células procariotas [17] no necesitamos contar con otras funciones que proporcionan otros tipos de alineadores.

Bowtie genera ficheros de salida en formatos Sequence Alignment/Map (SAM)/ Binary Alignment/Map (BAM), con las alineaciones, siendo estos en formato texto y binario respectivamente.

2.1.2. Conteo: HTSeq

HTSeq es una librería de Python que permite el análisis de datos de secuenciación de alto rendimiento High Throughput Genome Sequencing (HTGS). Uno de sus usos es el conteo de genes mediante la herramienta *htseq-count*. Esta herramienta realiza el conteo de genes a partir de los ficheros SAM/BAM (sección C.1.2) y un fichero General Feature Format (GFF)/General Transfer Format (GTF) con las posiciones de los genes en el genoma (sección C.1.3); esto es, cuántas alineaciones se solapan con cada uno de los genes anotados en el fichero GFF/GTF. Antes del uso de *htseq-count* es necesario ordenar e indexar las alineaciones utilizando alguna herramienta para tal efecto como puede ser Samtools (véase apéndice D). Este conteo es lo que se utiliza más tarde para determinar el nivel de expresión diferencial de un gen mediante los métodos proporcionados por herramientas como DESeq, que se explicará a continuación.

El método de conteo de *htseq-count* está especialmente diseñado para el análisis de expresión diferencial, por lo cual sólo son consideradas las lecturas que únicamente tienen coincidencias con un gen [18]. Las alineaciones que coinciden con distintos genes, o tienen solapamientos en diferentes posiciones son descartadas dependiendo del modo de conteo seleccionado. Esto se debe a que se quieren evitar los casos en los que genes con similitudes en su secuencia de nucleótidos puedan dar lugar a errores en la cuantificación de su expresión, aumentando así el número de falsos positivos en el análisis de la expresión diferencial.

La figura F.4 del apéndice F ilustra el funcionamiento del algoritmo implementado por la herramienta *htseq-count*. En el caso de los solapamientos el algoritmo funciona como se describe a continuación. Para cada posición i de la lectura, se crea un conjunto $S(i)$ para $i = 0, 1, ..., n$ el cual define el conjunto de todos los genes que se superponen en esa posición de la lectura. A partir de este conjunto, *htseq-count* proporciona tres tipos diferentes de conteo:

- **Unión**: devuelve la unión de todos los conjuntos $S(i)$ asociados a la posición i de la lectura.
- **Intersección estricta**: devuelve la intersección de todos los conjuntos $S(i)$. Puede darse el caso de la intersección con un conjunto $S(i) = \emptyset$.
- **Intersección no vacía**: devuelve la intersección de todos los conjuntos $S(i)$ no vacíos.

2.1.3. Análisis de expresión diferencial: DESeq

DESeq es un paquete de *Bioconductor project* de R [19] que proporciona una potente herramienta de análisis de expresión diferencial a partir de los datos obtenidos mediante el conteo de genes. Lo que se pretende con este tipo de análisis es determinar si para un gen dado se observa una diferencia considerable entre el número de veces que se expresa en una condición experimental frente a una condición de control.

Para realizar este tipo de análisis se ha recurrido a la estadística, modelizando la distribución del número de cuentas por cada gen de acuerdo a distintas distribuciones estadísticas. Inicialmente se consideró la distribución multinomial, aproximada por una distribución de Poisson, sin embargo, se constató que esta distribución producía resultados con variaciones más pequeñas de lo que se observaba en los datos. Para resolver este problema se recurre al modelo de la distribución binomial negativa *(Binomial Negativa (BN))*, la cual está determinada por la media μ y la varianza σ^2, contraponiéndose a la distribución de Poisson que está unívocamente determinada por la media μ [20]. Este análisis se basa en asumir que el número de lecturas en una muestra j que están asignadas a un gen i se pueden modelar con una distribución binomial negativa [20].

$$K_{ij} \sim NB(\mu_{ij}, \sigma_{ij}^2)$$

DESeq tiene como entrada una matriz $n \times m$, donde $i = 1 \ldots n$ recorre los genes y $j = 1 \ldots m$ indexa las muestras analizadas. Además, permite trabajar con experimentos sin réplicas.

Los resultados de DESeq vienen dados en formato de tabla en los que las columnas se corresponden con el nombre del gen, la media agregada del conteo de genes de las dos condiciones experimentales comparadas, la media desagregada del conteo de la primera condición experimental y de la segunda condición, el *fold change* [1], y su logaritmo, el p-valor del test estadístico de contraste de hipótesis y el p-valor ajustado para controlar la tasa de falsos positivos FDR [21].

[1] ratio de la media de conteo de la primera condición experimental frente a la segunda

2.2. Interfaces y aplicaciones

Para realizar el análisis de expresión diferencial se han desarrollado algunas interfaces y aplicaciones que integran diferentes herramientas del *pipeline* de análisis de datos RNA-seq y permiten el uso de las mismas sin necesidad de utilizar comandos UNIX. A continuación nombraremos algunas de ellas.

2.2.1. Galaxy

Galaxy [22] es una aplicación web que permite el uso de una gran variedad de herramientas de bioinformática. Es extensible, lo que permite que los programadores puedan integrar casi cualquier herramienta de análisis de secuencias. Fue creada como una forma de trabajo con ficheros de texto como por ejemplo secuencias de ADN, pero los nuevos desarrollos han añadido visualización de datos e interfaces para las herramientas. La mayor ventaja de Galaxy es la posibilidad que ofrece de que distintos investigadores puedan trabajar sobre los mismos datos vía web. Además permite compartir conjuntos de datos y *workflows*, permitiendo así la replicación de los experimentos y la comparación de los resultados. Otra de las funcionalidades que ofrece es editar y guardar *workflows* para su uso en un futuro. Galaxy dispone de servidores públicos pero también se puede descargar y ejecutar de forma local o en la nube, lo que permite usar almacenamiento adicional y recursos para la computación.

Tiene la desventaja de ser una aplicación tan completa que su complejidad es muy alta. Proporciona casi la totalidad del abanico de herramientas para diferentes estudios y análisis con genomas, lo cual obliga al usuario a conocer y discriminar entre todas las herramientas ofrecidas. Además, no proporciona visualización de los resultados en formato de gráficos.

2.2.2. Otras herramientas

Otras interfaces son Rockhopper [23] y RNASeqGui [24]. RNASeqGUI cuenta con la desventaja de ser una herramienta de difícil usabilidad lo cual no la hace atractiva para el usuario no experto. Por otro lado, Rockhopper es mas sencilla de usar, pero no incorpora herramientas estándar de *facto* lo que puede ser una desventaja a la hora de decidir si usarla.

3

Definición del proyecto

En este capítulo se procederá a definir y explicar brevemente el alcance del proyecto, así como la metodología escogida para el desarrollo del mismo y las herramientas utilizadas a lo largo del trabajo.

3.1. Alcance

La interfaz gráfica a diseñar e implementar tiene como objetivo gestionar y facilitar el análisis de expresión diferencial de genes RNA-seq mediante una interfaz gráfica de usuario. Se pretende, integrar las herramientas necesarias para el procesamiento de los datos del análisis RNA-seq en una única aplicación y ofrecer una automatización de la representación gráfica de los resultados obtenidos.

Sin embargo, conviene recalcar que no es el objetivo de esta aplicación ofrecer todas las herramientas disponibles en el mercado, si no realizar una selección de las herramientas más utilizadas para así facilitar a un usuario no experto el estudio de la expresión diferencial. Además, no se pretende, a nivel del presente trabajo, dar soporte a todos los formatos de datos que se pueden utilizar para este análisis, si no una vez más, decidir un formato y aplicar el análisis de acuerdo al mismo.

El proyecto incluye la creación de un contador de genes, el cual sirve de enlace entre la alineación del genoma y el análisis de expresión diferencial, definiendo un método de conteo de genes basado en el previo estudio de otros contadores tales como HTSeq [18]. Esta decisión ha sido tomada debido a la gran complejidad en el uso de la herramienta *htseq-count* y para mejorar el conocimiento y comprensión de la totalidad del análisis de expresión diferencial. El contador es la herramienta central del análisis. Esto obliga a conocer en detalle tanto los pasos previos al conteo como el análisis posterior. Se corresponden a la alineación y ordenación y al uso del DESeq respectivamente.

3.2. Metodología

Se ha optado desarrollar la aplicación con una metodología en cascada con retroalimentación tal y como se muestra en la figura 3.1. A continuación se presenta el proceso de desarrollo de software, en el cual se observa que se ha decidido no comenzar con una fase hasta no haber completado la anterior, sin embargo algunas de las subtareas sí que se han realizado de forma simultánea debido a su alta coorelación. Se ha descartado el modelo de desarrollo iterativo y el de espiral debido al corto periodo de tiempo en el que se pretende realizar el proyecto, pero la retroalimentación nos permite volver a pasar por algunas fases anteriores en caso de que sea necesario realizar modificaciones.

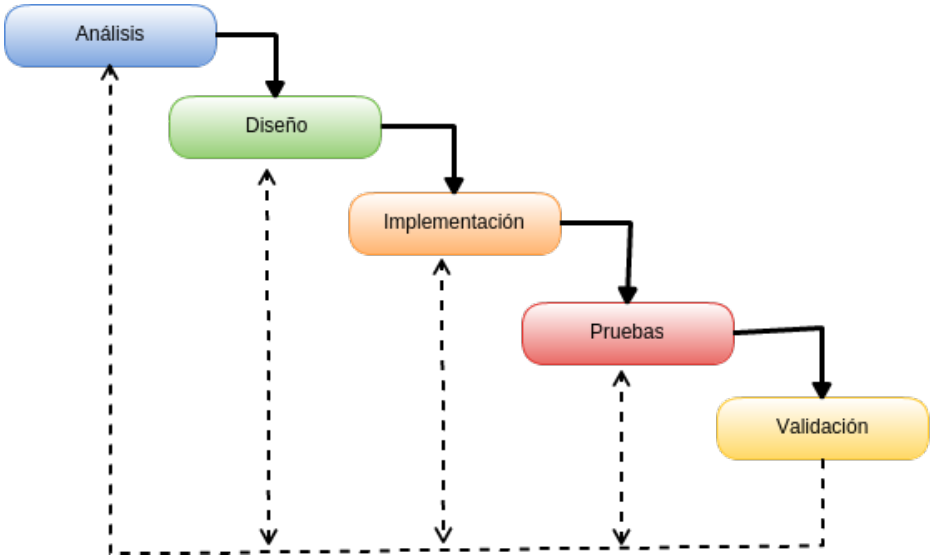


Figura 3.1: Metodología en cascada

Además, debido al extenso periodo de estudio y adquisición de conocimientos relacionados con la bioinformática se hace presente la necesidad de adoptar una metodología ágil de implementación. El tutor se ha implicado a la hora de revisar la solución de cada proceso del desarrollo así como incrementar la funcionalidad, siendo estos puntos las principales fuentes de la retroalimentación. Las etapas de cada iteración de la cascada consta de:

- El análisis consta de dos ramas, referentes al estudio del estado del arte y del proyecto que se quiere realizar:
  1. Análisis del estudio del arte sobre bioinformática, datos a analizar y herramientas que se quieren incluir en la aplicación.
  2. Análisis de los requisitos funcionales y no funcionales que se puedan establecer hasta el momento de la iteración para el desarrollo de la interfaz gráfica de usuario.
- Diseño de los módulos en los que se dividen los elementos del proyecto, creación de *workflow* de la aplicación y diseño de la interfaz para la funcionalidad establecida.
- Implementación de la funcionalidad para llevar a cabo el *workflow* del análisis integrado en la interfaz.
- Pruebas de la funcionalidad implementada y análisis de los resultados
- Reunión con la tutora para validar del proceso realizado y acordar nueva fase de desarrollo en caso de ser necesario añadir funcionalidad o revisar los requisitos.

La figura E.1 del apéndice E representa la distribución temporal de las tareas (ver tabla 3.1) y sus dependencias en un diagrama de Gantt. También se ha incluido una tabla (ver figura E.2) con las fechas y las horas dedicadas a cada una de las fases mostradas en el diagrama de Gantt.

3.3. Herramientas

Para la realización de este proyecto se ha hecho uso de herramientas para:

- Programar la interfaz gráfica de usuario.
- Probar la aplicación.
- Disponer de un control de versiones.
- Realizar copias de seguridad en la nube de los datos de la aplicación.
- Diseñar diagramas y maquetas.

3.3.1. Entorno de programación y librerías

El Integrated Development Environment (IDE) utilizado para la programación de la GUI ha sido RStudio [25] ya que nos permite la ejecución *in situ* del código así como la representación de gráficas y otros resultados. Desde ella se han podido integrar las diversas librerías (paquetes de R) necesarias para la implementación de la interfaz y las herramientas del RNA-seq desarrolladas en R [19] como es en este caso DESeq [20]. El lenguaje de programación R a este nivel no forma parte del plan de estudio del Doble Grado, por lo que ha sido necesario realizar un estudio y aprendizaje del lenguaje de programación y de las librerías utilizadas para este proyecto.

La interfaz se ha diseñado utilizando shinyFiles [26], shinydashboard [27] y Shiny [28]. Otras librerías de R utilizadas son ggplot2 [29], reshape, matrixStats [30] y DESeq.

Para la implementación del contador y herramientas para analizar las calidades de los *reads* de la secuenciación se ha utilizado el lenguaje C debido a su buen rendimiento y el gran tamaño de datos a manejar. Se ha utilizado el entorno de Sublime Text [31].

3.3.2. Almacenamiento y control de versiones

Como método de almacenamiento se ha decidido utilizar Dropbox y OneDrive, ambos, servicios de almacenamiento de archivos multiplataforma en la nube. Con esto se ha conseguido almacenar y sincronizar archivos en línea y entre ordenadores, para así compartir archivos y carpetas con otros, como es en este caso la tutora. Se han utilizado dos porque OneDrive permite la edición online de documentos y como método de seguridad para almacenar dos copias del trabajo con el fin de poder minimizar el impacto de un borrado accidental de archivos.

3.3.3. Diseño de diagramas y maquetas

Todos los diagramas y maquetas se han diseñado con Cacoo [32]. Las tablas incluidas en esta memoria se han creado con un generador automático de tablas online para LaTeX[33].

Fase	***Entrada***	***Tareas***	***Salida***
Estudio del problema		Análisis del problema biológico de la secuenciación genómica y la expresión diferencial. Estudiar los procesos de análisis y las herramientas utilizadas. Estudiar el paquete Shiny de R.	Pipeline del análisis de expresión diferencial.
Análisis	Pipeline	Definir los objetivos de la aplicación. Especificar los requisitos funcionales y no funcionales.	Definición, objetivos y alcance del proyecto. Relación de requisitos.
Diseño y arquitectura	Requisitos	Diseñar la arquitectura y módulos a implementar. Diseñar la interfaz web.	Arquitectura de la aplicación. Diagramas de diseño. Maquetas de la interfaz web.
Implementación	Información sobre la arquitectura y el diseño de la aplicación. Maquetas de la interfaz web.	Codificación de las herramientas y procesamiento del modelo de datos. Codificación de la interfaz gráfica de usuario y el controlador.	Código de la aplicación. Documentación del código de la aplicación.
Pruebas	Código de la aplicación Datos simulados Datos reales de expresión diferencial	Realizar el plan de pruebas. Evaluar la aplicación final.	Código de la aplicación validado y verificado.
Documentación	Conocimiento del estudio del problema y herramientas. Diseño, implementación y pruebas de la aplicación.	Documentar el proceso del proyecto y su desarrollo. Realización de la presentación para la defensa del TFG.	Memoria del Trabajo del Fin de Grado.

Tabla 3.1: Tabla de tareas

4
Requisitos

En este capítulo se enumerarán los requisitos de la aplicación a desarrollar. Para la elaboración de la lista de requisitos se ha realizado un análisis del problema: diseñar una interfaz gráfica que permita a usuarios no expertos en informática y/o bioinformática realizar un análisis de expresión diferencial de genes así como obtener una visualización sencilla y directa de los datos analizados y los resultados obtenidos en el análisis. Este análisis se ha realizado mediante la consulta a potenciales usuarios de la aplicación y la evaluación del estado del arte.

Los requisitos se agrupan en dos clases diferenciadas: funcionales y no funcionales. Los primeros describirán el comportamiento de la aplicación, mientras que los segundos describirán los parámetros de calidad del sistema.

4.1. Requisitos funcionales

Los requisitos funcionales que deberá cumplir la aplicación a desarrollar son los siguientes:

RF.1 Permitirá la navegación entre las diferentes pantallas de la aplicación, las cuales se van modificando en función de los datos introducidos por el usuario en la aplicación.

RF.2 Los usuarios de la aplicación podrán ejecutar/invocar cada una de las etapas del análisis de datos RNA-seq (alineación, conteo y análisis de expresión diferencial) siempre y cuando se haya procedido correctamente con los pasos anteriores del análisis.

RF.3 Permitirá determinar las rutas a las herramientas que se utilizarán para el análisis de expresión diferencial.

RF.4 Los formatos de fichero que se aceptarán serán:
Genoma de referencia: FASTA
Lecturas de genoma: FASTQ
Anotación de Genes: GFF
Alineaciones: SAM
Tabla de genes donde las filas representan cada gen y las columnas cada una de las réplicas

de las condiciones experimentales proporcionadas: fichero de texto. Para mas información sobre estos formatos ver Apéndice C.

RF.5 Permitirá establecer el directorio de trabajo donde el usuario quiere obtener los resultados de la aplicación.

RF.6 Permitirá la subida de ficheros que se encuentren almacenados en el PC desde que se trabaja. Los ficheros que se podrán subir son FASTA, FASTQ, y GFF.

RF.7 Generará un sistema de directorios en el directorio de trabajo seleccionado en el que se almacenarán los datos generados por la aplicación.

RF.8 Permitirá la manipulación de ficheros FASTA, para la generación de índices mediante la herramienta *bowtie-build*.

RF.9 La aplicación se encargará de hacer uso de los datos de índices generados a lo largo del proceso de análisis RNA-seq, evitando así la necesidad del usuario de volver a acceder a dichos datos.

RF.10 Será necesario establecer una condición experimental como condición de control y asignarle un identificador (cadena de caracteres). Para esta condición experimental se podrá indicar el número de replicas a analizar.

RF.11 Permitirá determinar el número de condiciones experimentales positivas diferentes de la de control de las que se quiere hacer un análisis de expresión diferencial frente a la condición de control. Para el correcto funcionamiento de la aplicación es necesario disponer de al menos una condición experimental diferente de la de control. Además cada condición experimental deberá ser acompañada por un nombre que la identifique.

RF.12 Para cada una de las condiciones experimentales, a parte de la de control, también se podrán determinar el número de réplicas que se proporcionarán.

RF.13 Para todas las réplicas se podrán de determinar los parámetros con los que se realizarán las alineaciones utilizando Bowtie. Es importante recalcar que los parámetros serán iguales para todas las réplicas. Los parámetros de Bowtie que proporcionará la aplicación son:

- **v** - Parámetro que indica el número máximo de diferencias entre las lecturas el genoma de referencia.
- **m** - Indica que no se reporten como alineaciones los *reads* que hayan alineado en más de m lugares del genoma de referencia.
- **p** - Número de hilos que se lanzan en paralelo.
- **x** - Distancia máxima entre los *reads* emparejados, *paired-end reads*.

RF.14 Para cada una de las réplicas se podrá determinar si se realizará una alineación *single-end* o *paired-end*, en cuyo caso habrá que proporcionar a la aplicación un fichero FASTQ o dos, respectivamente.

RF.15 Para cada una de las réplicas se generará un fichero de alineaciones en una subcarpeta del directorio de trabajo y su nombre vendrá determinado por el identificador dado a la condición de experimental y el número de réplica. Estos ficheros se generarán mediante la ejecución de la herramienta Bowtie.

RF.16 El formato de salida es el formato SAM (para mas información ver Apéndice C).

RF.17 Proporcionará visualizaciones de los resultados generados por el la herramienta Bowtie en formato de gráficas y tablas. Por cada condición experimental y cada réplica, porcentajes de alineación, no alineación, alineación múltiplen y alineaciones descartadas.

RF.18 Se mostrarán gráficas con el promedio y desviación típica de las calidades de las lecturas o q-scores (véase Apéndice C) en función a la posición de cada nucleótido en la lectura.

RF.19 El usuario podrá realizar la ordenación de los ficheros SAM generados para el caso de que se desee continuar el análisis con la herramienta HTSeq.

RF.20 El usuario podrá como realizar el conteo de genes, usando o bien la herramienta HTSeq o la herramienta desarrollada en este trabajo, ContadorTFG.

RF.21 Se permitirá la subida de un fichero GFF con las referencias a los genes (para más detalles ver Apéndice C).

RF.22 En cada una de las herramientas de conteo se podrán modificar los parámetros de ejecución. Estos parámetros son:

- **Strand** - Indica si las lecturas son *stranded* o no.
- **Overlaping** - Porcentaje de solapamiento entre el gen y la alineación.
- **Feature** - Nombre de la característica que se quiere contar.
- **Orden** - Por nombre o posición.
- **Modo** - En el caso del HTSeq se ofrecen los modos Unión, Intersección estricta e Intersección no vacía.

RF.23 El usuario podrá visualizar la matriz de conteo generada en modo de tabla. Se podrá ordenar por orden alfabético de los genes o por cualquiera de las columnas. Se podrán realizar búsquedas.

RF.24 Se ofrecerá una visualización gráfica de los datos obtenidos mediante el conteo. Se mostrará una gráfica por cada condición experimental no de control. Será un diagrama de barras que representará en diferentes colores la condición experimental de control y la de no control. Además cada réplica se corresponderá con una barra. El valor numérico de cada barra también se mostrará. Por último se podrá seleccionar el gen del que se quiere obtener la información del conteo.

RF.25 La ejecución del contador se realizará mediante la pulsación de un único botón para todas las réplicas proporcionadas generando un fichero de salida que se corresponderá con la matriz de conteo necesaria para la realización del análisis de expresión diferencial de genes. Este fichero se almacenará en una de las subcarpetas del directorio de trabajo.

RF.26 Se podrá realizar el análisis de expresión diferencial de cada una de las condiciones experimentales que se hayan proporcionado frente a la condición de control utilizando la herramienta DESeq.

RF.27 Se podrán configurar los parámetros a utilizar para el análisis de expresión diferencial. Estos parámetros son:

- **FDR** False discovery rate [21].
- **Metodo para la expresión diferencial** Modo de calcular la dispersión empírica.

- **Modo de uso compartido** Selección de la dispersión empírica de entre las dos calculadas para cada gen.
- **Tipo de ajuste** Ajuste de relación media-dispersión.

RF.28 Se mostrarán los resultados del análisis de expresión diferencial llevado a cabo con DESeq en formato de gráficas y tablas. Los gráficos que se proporcionarán se describen en la sección 5.5.3.

RF.29 Los resultados generados mediante la herramienta DESeq se almacenarán en el directorio de trabajo, en un fichero de texto denominado resultados.

4.2. Requisitos no funcionales

Los requisitos no funcionales que deberá cumplir la aplicación a desarrollar son los siguientes:

RNF.1 La aplicación va dirigida a la comunidad de científicos con poca experiencia en informática y/o bioinformática que necesiten realizar análisis de expresión diferencial genómica con células procariotas.

RNF.2 El sistema mostrará de forma precisa la pantalla en la que se encuentra el usuario en cada momento, así como las acciones que puede realizar.

RNF.3 La aplicación mostrará un menú lateral que permitirá navegar por las pantallas de la interfaz. Este menú se podrá minimizar.

RNF.4 Toda acción del usuario en la interfaz gráfica tendrá una respuesta visual que indique el resultado de la misma.

RNF.5 La interfaz gráfica será una web y podrá ser visualizada en distintas resoluciones de pantalla con una experiencia de usuario similar.

RNF.6 Será compatible con un sistema UNIX Linux 14.04 que soporte las herramientas que utiliza la aplicación para el análisis RNA-seq así como una versión de R 3.2.2.

RNF.7 La interfaz gráfica mostrará los textos en ingles.

RNF.8 No se garantiza un tiempo límite de las ejecuciones de herramientas y generación de gráficas que pueden suponer un alto coste computacional dependiendo de la naturaleza de los datos a analizar.

RNF.9 Se incluirá una barra de progreso que determina el proceso de ejecución de algunas de las herramientas.

5 Diseño

En este capítulo se describe el diseño de la aplicación a desarrollar. Tras el análisis de requisitos desarrollado en el capítulo 4, se ha decidido dividir el diseño de la aplicación en dos partes principales: la interfaz de usuario y el motor de análisis.

5.1. Patrón de la arquitectura

Se ha seguido el patrón de arquitectura MVC, Modelo-Vista-Controlador, que separa las herramientas y los datos a utilizar (modelo y modelo de datos), la interfaz gráfica (vista) y la modificación de la misma y control de la ejecución del análisis RNA-seq (controlador).

En la figura 5.1 se puede ver la interacción entre cada una de estas tres partes así como con el usuario final de la aplicación.

5.2. Modelo: El motor de análisis

El motor de análisis se encarga de procesar los datos utilizando para ello las herramientas descritas en el Capítulo 2. El usuario no tiene acceso directo a este motor, si no que utilizará la interfaz gráfica para facilitar los datos y es la misma aplicación la que, a partir de este momento, se encarga de gestionarlos mediante el controlador y su interacción con el modelo.

El proceso de análisis que se ha diseñado para esta aplicación se muestra en la figura F.1 del apéndice F. Se puede observar que se siguen los procesos generales que se muestran en el *workflow* representado en la figura F.2.

Este motor se compone de una herramienta de creación de índices y alineación de lecturas a un genoma de referencia, la cual se ha decidido que sea Bowtie. Como se muestra en la figura F.1 se puede ver como primeramente se deberán crear los índices del genoma de referencia. A continuación se procede a realizar las alineaciones de las lecturas de los experimentos a un genoma de referencia, utilizando los índices previamente generados. Una vez más, el usuario proporciona cada una de las condiciones experimentales y las réplicas de cada una de ellas a

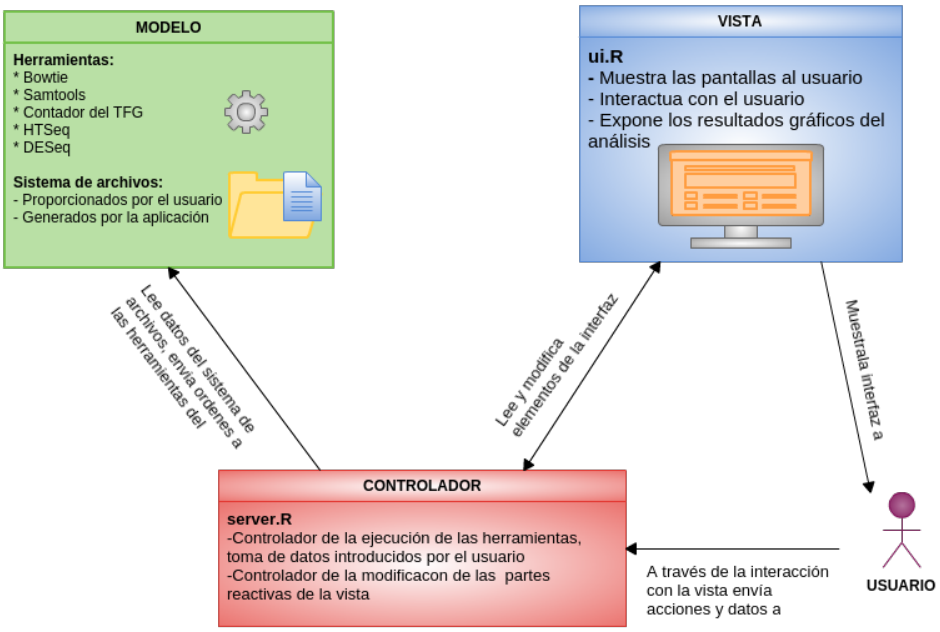


Figura 5.1: Modelo-Vista-Controlador (MVC)

través de la interfaz gráfica. Los datos necesarios son FASTA y FASTQ cuyo formato se describe en el la sección C.1.1 del apéndice C.

Una vez generados los ficheros de alineaciones SAM, cuyo formato se ha explicado en C.1.2, se puede proceder a realizar el conteo de genes, lo que se podrá llevar a cabo con dos herramientas diferenciadas. Por una parte se ha implementado un contador, el cual se explicará en 5.2.1 con más detalle y por otra parte se ha decidido proporcionar la opción de utilizar la herramienta HTSeq, para la cual es necesario hacer uso de alguna de las funciones que proporciona Samtools. A partir del los datos GFF y las alineaciones se genera la matriz de conteo. Los formatos de los datos se describen en C.1.3.

Una vez obtenida la matriz de conteo asociada a cada condición experimental se procede a hacer uso del módulo DESeq [20]. Para ello se han seleccionado las funciones que nos ofrece este paquete y que nos permiten realizar los siguientes estudios:

- Estimación de los factores de normalización. Dado que los experimentos no tienen por que contener el mismo número de lecturas y alineaciones, es necesario normalizar los datos obtenidos en la matriz de conteo.
- Estimación de la dispersión de los datos y calcular la transformada de estabilización de la varianza.
- Análizar la expresión diferencial de genes asumiendo que los datos de conteo siguen una distribución binomial negativa.
- Ajustar los p-valores obtenidos a partir del test estadístico de contraste de hipótesis sobre la binomial negativa utilizando la aproximación de Benjamini [21], que permite controlar la tasa de falsos positivos FDR (*False discovery rate*) de un tanto por ciento.
- Realizar el conteo del número de genes diferencialmente expresados.

- Inspeccionar el resultado de los *p-valores*, mediante un gráfico en diagramas de puntos y con un histograma de los *p-valores ajustados*. En el diagrama de puntos se muestran aquellos genes que están diferencialmente expresados, mediante el *fold change* en valor logarítmico, y en el histograma se representa el número de genes cuyo p-valor ajustado esté por debajo del FDR.

5.2.1. Contador

Como se ha mencionado al comienzo de la sección 5.2, se ha decidido diseñar un contador de genes que proporcione una funcionalidad similar a la que ofrece HTSeq, uno de los estándares de *facto* para el conteo de expresión génica. HTSeq es una herramienta difícil de utilizar y de hacer funcionar, debido a su amplio espectro de versiones su complejo rango de comandos de ejecución.

A continuación se expone el diseño del contador que se ha incluido en esta aplicación, con el fin de proporcionar un conocimiento mas profundo del funcionamiento de este tipo de análisis y ofrecer un modo de conteo sencillo.

Una de las diferencias con HTSeq, es que esta herramienta permite el análisis simultáneo de todos los ficheros de alineaciones y no requiere el procesamiento previo de los datos generados por Bowtie evitando así el uso de una herramienta adicional, en este caso la herramienta Samtools. Además, el contador implementado en el trabajo genera una matriz de conteo completa con una columna por cada experimento, mientras que HTSeq genera una matriz por cada experimento haciendo necesario el procesamiento de la salida de HTSeq para unificar todos los ficheros en una única matriz que será la entrada de la herramienta DESeq.

La gran diferencia con HTSeq es el modo de conteo. En la figura F.3 del apéndice F se muestran los casos en los que se considera que una lectura a alineado con un gen u otras características que se quieran estudiar teniendo en cuenta que la superposición supere un tanto por ciento de la lectura, determinado por el usuario.

El programa trabaja con dos tipos de estructuras, las cuales se utilizan para almacenar en cada uno de los tipos de datos que se necesitan para realizar el conteo. Estos son, las alineaciones de los ficheros SAM y la información de los genes anotados en el fichero GFF. La estructura deberá almacenar todos los campos definidos en C.1.3 para los GFF y en C.1.2 para las alineaciones. Además, se ha decidido crear un módulo para el análisis de cada uno de estos tipos de datos haciendo así mas sencilla su implementación. A continuación se exponen las funciones de las que constará cada uno de los módulos:

Módulo 1: Procesamiento de los ficheros SAM

- *split_SAM*(): esta función se encarga de leer cada uno de los campos de las alineaciones y almacenarlos en la estructura.
  - Input: delimitador de los campos, alineación.
  - Output: estructura del SAM, como parámetro. 0 si todo ha ido bien, -1 si error.
- *create_list_SAM*(): se encarga de crear una lista enlazada de estructuras de las alineaciones en el caso de lecturas en parejas (*paired-end reads*). En el caso de lecturas únicas (*single-end reads*), se construye un único nodo de esta estructura.
  - Input: cadena con una alineación del fichero SAM.

  - Output: Puntero a una estructura SAM.

- *valid_align*(): función encargada de determinar si se ha producido una alineación o no, dado que el fichero SAM puede contener también información de lecturas que no han alineado con el genoma de referencia.

  - Input: Estructura SAM, el *strand* en caso de querer discriminar las alineaciones, y un identificador para indicar si se trata de lecturas *single-end* o *paired-end*.
  - Output: True en caso de ser una alineación válida, False en caso contrario.

- *read_file_SAM*(): con ella se leerá el fichero de datos de las alineaciones.

  - Input: fichero SAM, puntero a la lista enlazada de GFF, indicador de *paired-end*, *single-end*, indicador del *strand* y el número de réplicas.
  - Output: 0 en caso de que se realice correctamente, -1 en caso de error.

- *free_SAM*(): esta función servirá para liberar los datos SAM almacenados durante la ejecución del programa.

  - Input: puntero a la estructura SAM que se quiere liberar.

Módulo 2: Procesamiento de los ficheros GFF

- *split_GFF*(): con esta función se leerán y almacenarán la información correspondiente a un gen anotado y contenida en el fichero GFF.

  - Input: delimitador de los campos en el fichero GFF, referencia al gen, número de réplicas, separador del nombre del gen en el fichero GFF.
  - Output: estructura que compondrá la lista de GFF.

- *create_list_GFF*(): se encarga de crear una lista enlazada de estructuras de los genes con la información contenida en el fichero GFF.

  - Input: línea del fichero GFF, separador del nombre del gen.
  - Output: puntero a la lista enlazada de estructuras GFF.

- *free_GFF*(): esta función servirá para liberar la lista de estructuras de los genes.

  - Input: Primer elemento de la lista GFF.

- *read_file_GFF*(): se encarga de la lectura del fichero GFF con información sobre los genes anotados del genoma de referencia.

  - Input: fichero GFF, puntero al primer nodo de la lista GFF, separador del nombre del gen, nombre del *feature* que se quiere contabilizar.
  - Output: puntero a la lista enlazada de GFF.

Módulo 3: Conteo

Este módulo consta de la función más importante del diseño del contador, donde

- *counting_alignments*(): se realiza el conteo de alineaciones por cada gen del fichero GFF de referencia.
  - Input: como entrada un puntero a la estructura SAM, puntero a la lista enlazada de GFF, número de réplicas, porcentaje de solapamiento.
  - Output: estructura GFF con el número actualizado de cuentas de los genes.
- *print_counts*(): se utilizará para mostrar los resultados del conteo.
  - Input: lista enlazada de genes y el número de réplicas.
  - Output: fichero con el conteo de genes.

5.3. Vista: Interfaz de Usuario

La interfaz se ha diseñado basándose en el *framework* y bibliotecas Shiny [34], Shinydashboard [35] y ShinyFiles [26] de R las cuales proporcionan un conjunto de componentes y métodos que permiten la creación de una interfaz sencilla, interactiva y de fácil uso para un usuario no experto. Gracias a Shinydahsboard, la interfaz estará dividida en una cabecera, un menú lateral y una pantalla principal.

Cabecera: Incluirá el nombre de la aplicación así como la opción de ocultar o mostrar el menú lateral.

Sidebar: Inclirá un menú listando los pasos del análisis, si es necesario desplegables, para poder mostrar cada uno de las subtareas u opciones de cada paso. Cada uno de los items del menú se corresponde a uno de los procesos del *workflow* mostrado en la figura F.1. El menú puede ocultarse en cualquier momento y volver a desplegarse, en caso de ser necesario para una mejor visibilidad de la ventana principal.

Pantalla principal: Se trata de una ventana interactiva, la cual se compone de pantallas asociadas a cada uno de los ítems y subítems del menú. Cada una de las pantallas se compone de *boxes*, *tabboxes*, *widgets* y botones a través de los cuales el usuario interactúa con la aplicación. Además, por cada ítem del menú se proporciona una pantalla con información sobre el uso de las mismas. Las páginas serán reactivas a las entradas del usuario, generando *boxes* con los *widgets* necesarios para que se puedan proporcionar los datos y ficheros que a continuación usará el controlador para crear y ejecutar las ordenes necesarias para la invocación de las herramientas del modelo.

5.3.1. Maquetas

M.1 En esta pantalla, mostrada en la figura G.1, se mostrará una visión general de la aplicación, haciendo una breve explicación de cada una de las herramientas que se utilizan a lo largo del análisis. Además, esta pantalla se encarga de preguntar al usuario la información necesaria para localizar los directorios de trabajo, así como los lugares en los que tiene alojadas las herramientas. Satisface los requisitos **RF.1**, **RF.5** y **RF.7** del capítulo 4.

M.2 Es la pantalla asociada a la explicación de la parte del análisis correspondiente a la herramienta Bowtie. Se mostrarán *boxes* para cada parte del análisis en las que se incluirá el texto explicativo o los pasos a seguir. Es una pantalla muy sencilla, por lo que no se proporciona una representación gráfica.

M.3 Pantalla asociada a la creación de índices con la herramienta Bowtie. La figura G.2 muestra esta maqueta. Permite determinar el nombre de los índices, proporcionar el fichero de referencia del genoma y muestra un botón mediante el cual se procede a la generación de los mismos. Satisface el requisito **RF.8** del capítulo 4.

M.4 En esta pantalla se produce la generación de alineaciones de los experimentos con el genoma de referencia. Se muestra en la figura G.3. Es en esta pantalla donde el usuario ha de introducir los metadatos. El usuario puede determinar el número de condiciones experimentales que se quieren analizar, el número de réplicas que se proporcionan por cada condición experimental y los nombres de las condiciones experimentales. Para la condición de control se pueden determinar el nombre y el número de réplicas. Además, para cada una de las réplicas proporcionadas se podrá determinar si son experimentos emparejados o individuales. Por otra parte, en esta pantalla se determinan los parámetros de ejecución de la herramienta Bowtie. Se proporciona un botón mediante el cual se ejecuta la herramienta Bowtie. Satisface los requisitos **RF.9**, **RF.10**, **RF.11**, **RF.12**, **RF.13**, **RF.14** y **RF.15** del capítulo 4.

M.5 En esta pantalla se muestran los resultados obtenidos tras la alineación. Estos datos se corresponden con las estadísticas generadas por la herramienta Bowtie a modo de diagrama de barras y tabla y los datos de calidad de las lecturas en un gráfico de errores representando la media y la dispersión por posición de la lectura. Se muestran en la figura G.4. Las gráficas que se proporcionan se detallan en la sección 5.5.1 y se pueden ver en las figuras G.9,G.10 y G.11. Satisface los requisitos **RF.17** y **RF.18** del capítulo 4.

M.6 La ventana asociada al item Samtools es la única de la aplicación que no es obligatorio usar. Sólo es necesaria en caso de que se quiera realizar el análisis con HTSeq. En esta pantalla se procesan los datos alineados y se procede a ordenar los datos por nombre o posición de las alineaciones. Para ello se podrá determinar que tipo de ordenación se desea realizar. La ejecución de esta herramienta generará archivos de alineaciones en formato SAM y BAM ordenados. La maqueta se muestra en la figura G.5. Satisface el requisito **RF.19** del capítulo 4.

M.7 En esta pantalla se proporciona una explicación detallada del uso de cada uno de los contadores que se proveen en la aplicación.

M.8 Para el contador desarrollado en el Trabajo de Fin de Grado (TFG) se ofrece una pantalla en la que aparecerá un bloque de opciones para cada condición experimental que se haya especificado en **M.4**, así como por cada una de las replicas existentes. El usuario encontrará los parámetros que puede determinar para la ejecución de la herramienta y podrá proporcionar el fichero de referencia GFF. La maqueta se puede ver en la figura G.6. Satisface los requisitos **RF.20**, **RF.21**, **RF.22**, **RF.25** y **RF.23** del capítulo 4.

M.9 En la ventana asociada a la herramienta HTSeq nos encontramos muchas similitudes con la maqueta anterior, modificando los parámetros que se pueden especificar. En este caso no es necesario especificar parámetros específicos para cada uno de los ficheros de alineación si no que con los parámetros son generales a todos ellos.

M.10 En esta pantalla se mostrará una tabla con pestañas en las que se podrán visualizar diagramas de barras representando el conteo de cada condición experimental frente a la condición de control. Se seleccionará el gen del que se quiere representar el resultado. Esto se muestra en la figura G.7. Además se muestran las tablas representando la matriz de conteo de cada condición experimental. Estas tablas se podrán ordenar por columnas. Satisface los requisitos **RF.23** y **RF.24** del capítulo 4.

M.11 En esta pantalla se realizará la interacción necesaria para la realización del último paso del análisis de expresión diferencial a partir de la herramienta DESeq. Se muestran los parámetros de ejecución del DESeq. En esta pantalla se mostrará un listado con todas las condiciones experimentales no de control proporcionadas por el usuario, de forma que se pueda elegir cual se va a analizar frente a la condición de control. Los resultados obtenidos mediante el análisis se mostrarán en esta misma pantalla a modo de ventana con pestañas para las diferentes gráficas y tablas. Esta maqueta se muestra en la figura G.8. Satisface los requisitos **RF.26**, **RF.27**, **RF.28** y **RF.29** del capítulo 4.

5.4. Controlador

En este caso el controlador se encargará de interactuar entre la interfaz gráfica y el modelo. Por ello su función será la de recoger los datos que el usuario proporcione mediante la interfaz y gestionarlos de forma que o bien se generen nuevos elementos de la interfaz, o bien se ejecuten las herramientas del modelo. Es el punto en el que se produce prácticamente toda la funcionalidad de la aplicación. En él se distinguen tres grandes tipos de control:

- Control de la interfaz: se encarga de renderizar y actualizar los elementos de la interfaz gráfica a partir de los *input* del usuario.
- Ejecución de las herramientas del análisis de expresión diferencial, previa obtención de datos: recoge los datos proporcionados por el usuario y los gestiona los datos del modelo para poder crear y ejecutar las ordenes de las herramientas.
- Generación de tablas y gráficos con los resultados: recoge los datos generados por las herramientas del análisis y los gestiona para proporcionar al usuario una visualización de los mismos a través de la interfaz.

5.5. Gráficas y resultados a mostrar

Los representación de los resultados en formato gráfico pretende proporcionar al usuario una lectura de los resultados rápida y sencilla, evitando así la necesidad de realizar tediosos estudios de los ficheros generados durante el análisis.

5.5.1. Bowtie

Por cada condición experimental, se muestran los resultados del análisis de Bowtie, representados en forma de tabla y diagrama de barras en los que se representan valores tanto en tanto por ciento como en valor numérico del número, tal y como se muestra en las figuras G.9 y G.10, respectivamente.

- Las lecturas procesadas.
- Las lecturas que han alineado con el genoma de referencia correctamente.
- Las lecturas para las que no se ha encontrado una alineación.
- Las lecturas que han sido descartadas.

Estos cuatro resultados se muestran para cada una de las réplicas proporcionadas para las condiciones experimentales y la condición de control. Además, se muestran los resultados de calidad de las lecturas, para lo cual se muestra la media y la desviación típica de la calidad de cada posición de las lecturas. Esta gráfica se muestra en la figura G.11.

5.5.2. Contador

Se muestran los resultados obtenidos mediante el conteo de genes, dando la opción al usuario a determinar el gen del cual quiere obtener la información, como se muestra en la figura G.12 (selección del gen del que se quiere obtener el conteo). Puesto que las muestras no tienen el mismo tamaño se normaliza la matriz de conteo, utilizando el mismo método que en análisis de expresión diferencial. Con ello se procede a mostrar el número de veces que se ha contado el gen seleccionado en cada una de las réplicas de una condición experimental y la condición de control. La gráfica muestra un diagrama de barras con los valores normalizados de una condición experimental junto a la condición de control y además se ofrece el valor original del conteo en valor numérico. Un ejemplo de lo que se pretende mostrar con esta gráfica se puede ver en la figura G.13.

5.5.3. DESeq

Gráficas y resultados del análisis de expresión diferencial. Se muestran los siguientes resultados:

- Gráfica de la estimación de la dispersión por gen con la dispersión media ajustada.
- Representación de la media del conteo normalizado (de ambas condiciones experimentales) frente al logaritmo del *fold change* (cuanto cambia el valor del gen respecto al valor inicial). Se representan en rojo aquellos genes que tengan un p-valor ajustado menor que el FDR establecido.
- Histograma de los p-valor ajustados y tabla con los genes diferencialmente expresados.
- Representación de los p-valor frente a logaritmo del *fold change*.
- Tabla con el análisis de los genes.

Las figuras G.14, G.15 y G.16 son ejemplos de los resultados mencionados.

6
Implementación

En esta sección se muestra la estructura del proyecto y se explica la implementación de los diferentes módulos que conforman el patrón MVC, Modelo-Vista-Controlador, expuesto en la sección 5.1. Gran parte de la información necesaria para la implementación de la vista y el controlador se ha obtenido de las páginas oficiales de RStudio donde se encuentran tutoriales y ejemplos de Shiny [34] y de Shinydashboard [35], así como de la documentación oficial de los paquetes Shiny [36] y Shinydashboard [37]. Para la implementación del contador, nos hemos basado en el estudio del artículo que propuso el contador HTSeq [18] y se ha decidido implementarlo en C. Por último, para llevar a cabo el análisis de expresión diferencial se ha resuelto implementar esta parte siguiendo el estudio de *Simon Anders et all.* en el artículo [4]. La implementación de las gráficas se ha realizado con las librerías *ggplot2* de R.

6.1. Contador

Para la implementación del contador se ha utilizado el lenguaje de programación **C**. La elección de C como lenguaje de programación viene motivada por la necesidad analizar conjuntos de datos bastante grandes y se alega a la eficiencia y rapidez del mismo para conseguir el fin propuesto. Otra de las grandes motivaciones del uso de C como lenguaje de programación para esta herramienta es la fácil integración de C en R, lenguaje que se ha utilizado para desarrollar la interfaz gráfica de usuario. Se ha seguido la siguiente estructura:

- *main.c*. Está implementado para que reciba un número variable de argumentos y a partir de ellos:

    1. Recoja los parámetros obligatorios para la ejecución del contador.
    2. Procese las muestras o réplicas de las condiciones experimentales que se quieren analizar.
    3. Haga uso de los módulos de análisis de cada uno de los tipos de datos recogidos en los argumentos
    4. Imprima los resultados obtenidos mediante el análisis.

```
typedef struct gff
{
        char * seqname;
        char * source;
        char * feature;
        int start;
        int end;
        float score;
        char * strand;
        int frame;
        char * name;
        int * cont;
        struct gff * next;

} GFF;
```

(a) Estructura GFF.

```
typedef struct sam {
   char * qname;
   int flag;
   char * rname;
   int pos;
   int mapq;
   char * cigar;
   char * rnext;
   int pnext;
   int tlen;
   char * seq;
   char * quality;
   struct sam * next;

} SAM;
```

(b) Estructura SAM.

Figura 6.1: Descripción de las estructuras de datos utilizadas en el contador.

- *gff.c* y su correspondiente *gff.h*. Incluye las funciones definidas en la sección 5.2.1 del diseño. Para ello se ha definido la estructura de datos que se muestra en la figura 6.1a.
- *sam.c* y su correspondiente *sam.h*. En este módulo se implementan las funciones definidas en el diseño 5.2.1 y definido la estructura 6.1b. Además, las funciones definidas para el módulo contador definido en la sección 5.2.1 se han incluido junto con las funciones de de procesamiento de ficheros SAM en esta parte de la estructura del contador.

6.2. Análisis de expresión diferencial

Para esta parte del proyecto se ha utilizado el lenguaje de programación R, en particular uno de los paquetes proporcionados por el proyecto *Bioconductor* llamado *DESeq* [20], una de las herramientas estándar de facto para el análisis de expresión diferencial [4]. El análisis realizado se muestra en el anexo D en la sección D.3. A partir de las funciones del paquete se ha implementado el análisis que se requería para este trabajo. El análisis de expresión diferencial incluye la obtención de datos, su procesamiento (normalización, ajuste a una binomial negativa y test estadístico) y, por último, la representación de los resultados en forma de gráficas y tablas (véase sección 5.5.3).

6.3. Generación de resultados gráficos

El motor gráfico se ha implementado para obtener los datos resultantes del cada una de las partes del análisis y generar las correspondientes gráficas con el fin de facilitar la lectura de los resultados más significativos. Para ello, el controlador se encarga de llamar a este módulo que consta de las funciones que generan las gráficas definidas en la sección 5.5. Para ello se ha utilizado el paquete *ggplot2* [29] de R a partir del cual se han representado los resultados.

Se ha decidido utilizar este paquete por el amplio rango de gráficas que proporciona y su fácil integración en la interfaz. Los elementos utilizados han sido:

- ggplot: inicialización de un objeto de tipo ggplot, en el cual se define el *data frame* que se va a representar. Se pueden determinar los datos que se quieren representar en los ejes.
- geom_ bar: diagrama de barras, bien simple o de representación múltiple. Se ha utilizado para representar los valores estadísticos de la ejecución de Bowtie, los valores de conteo de genes. Un ejemplo de estas gráficas se muestra en las figuras G.9 y G.13.
- geom_histogram: función para representar histogramas. Se ha utilizado para la representación de los resultados del análisis de expresión diferencial llevado a cabo con DESeq. Un ejemplo de esta gráfica se puede ver en la figura G.15.
- geom_vline: función que permite representar un segmento vertical en un valor determinado del eje de abscisas. Se utiliza para realizar la separación en genes diferencialmente expresados y no expresados en el histograma de p-valores ajustados. Este elemento se puede ver en la gráfica G.15.
- geom_point: utilizados para la representación de diagramas *scatterplot* tales como la representación de los resultados generados por DESeq para representar los valores del logaritmo del *fold-change*. En la figura G.14 se puede ver un diagrama de puntos.
- geom_line: diagrama de lineas, utilizado para representar los valores medios de calidad de los *reads* por posición. La gráfica G.11 es un ejemplo de ello.
- geom_errorbar: representación de los errores en un diagrama. En este caso se utilizan para la representación del diagrama de la media y desviación típica de las calidades de los *reads*. Las desviaciones típicas de la gráfica G.11 muestra un ejemplo.
- geom_text: permite añadir texto a las gráficas. Ha sido de gran utilidad pera mostrar los valores numéricos del conteo de genes sobre el diagrama de barras. El numero que se muestra en las barras de la figura G.13 se ha generado con esta función.

Estos elementos se integran dentro del objeto creado con ggplot, y se pueden superponer generando un único gráfico con distintos elementos. Además, esta librería permite la definición de las paletas de color y generación de leyendas. Esto ha permitido dotar de colores representativos a las gráficas, mediante agrupación por colores y datos representados.

Antes de utilizar las funciones de ggplot2 es necesario crear los *data frame* que permitan un buen uso de los datos, así como su previa manipulación para poder llegar a la representación deseada.

6.4. Interfaz gráfica

Shiny divide su funcionalidad en dos partes. Por un lado proporciona la vista de la interfaz mayormente a través de un módulo denominado *ui.R* y la generación de la funcionalidad se recoge en un módulo llamado *server.R*. En la figura 6.2 se muestra el modo en el que estos dos módulos se comunican y permiten dotar a la interfaz de un carácter dinámico.

La funcionalidad de Shiny que se a aplicado en el proyecto, basada en la Vista-Controlador definida en la sección 5.1, se traduce en la interacción entre los módulos *ui.R* y *server.R*. El

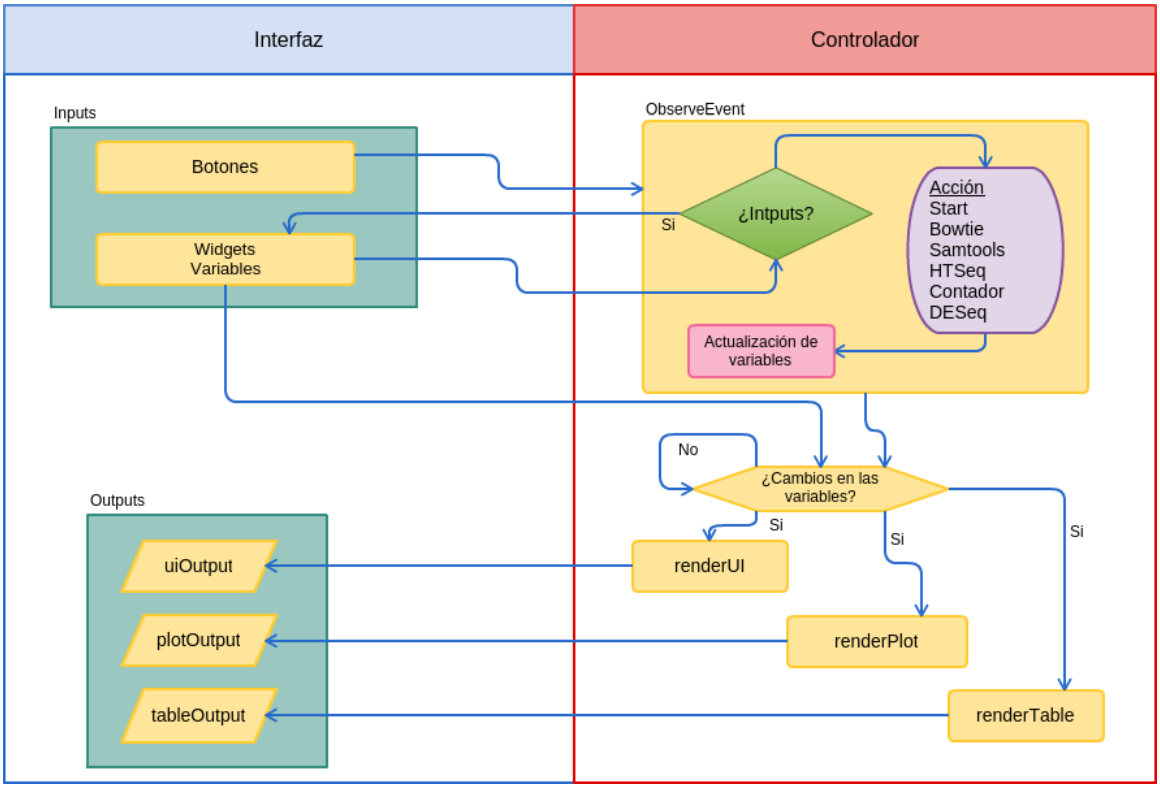


Figura 6.2: Funcionalidad de la vista y el controlador mediante Shiny

usuario interactúa con los elementos que se muestran en la interfaz, definidos en la lista 6.4.1, desencadenando las acciones de las funciones del controlador. Estas acciones pueden tener efectos sobre el modelo, mediante la ejecución de las herramientas del análisis, o bien sobre la vista de la interfaz, generando nuevos elementos para el usuario. La primera se recoge dentro de las funciones *observeEvent*, las cuales pueden desencadenar a su vez las funciones de actualización de la interfaz *renderUI*, *renderTable* y *renderPlot*. Una vez que estas funciones han finalizado el control vuelve a estar sobre el usuario que podrá seguir interactuando con la interfaz.

6.4.1. ui.R

Para la implementación de la interfaz gráfica se utilizado Shiny, un paquete de R que proporciona un potente *framework* para el desarrollo de aplicaciones web, y Shinydashboard para la creación de una interfaz con la estructura de un *dashboard*. Cada una de las pantallas de la aplicación se compone de los siguientes elementos:

- *Header*
  Este espacio es común a todas las pantallas e inmutable. En él se incluye el título o nombre de la aplicación, así como un botón que permite el despliegue u ocultación del *sidebar*.

- *Sidebar*
  Esta sección lateral se ha implementado a modo de menú, *sidebarMenu* (ver figura 6.3) en el cual se incluyen los *menuItem* y *menuSubItem*. Cada uno de estos item están asociados mediante un identificador a las pantallas de la aplicación. En el menú, Shiny considera que un item que tiene subitmens no ha de estar linkado a ninguna pantalla, por lo que

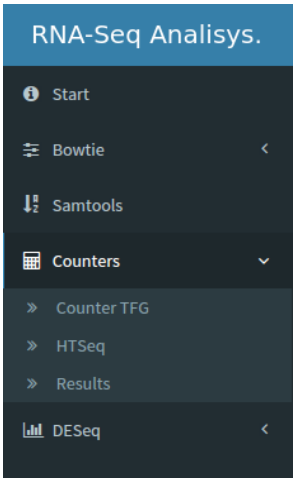


Figura 6.3: Menú lateral

desactiva la funcionalidad *on-click* asociada a la pantalla principal, y crea una nueva funcionalidad de despliegue de los subitems.

Puesto que se desea asociar una pantalla explicativa a cada uno de las herramientas de la aplicación, se ha recurrido a añadir la doble funcionalidad a los items con hijos, para que puedan tener una ventana asociada a ellos.

- *Body*
  El cuerpo principal de la aplicación se compone de todos los elementos necesarios para la transmisión de datos entre la aplicación y el usuario. A continuación se listan los elementos que se han utilizado en el proyecto para la entrada de datos.

  - radioButtons: elemento utilizado para la selección única de opciones. Ejemplificado en la primera caja de la figura B.5 en el apéndice B.
  - sliderInput: widget que permite establecer un valor numérico entre un rango de valores determinado en intervalos. Ejemplificado en la segunda caja de la primera fila de figura B.5 en el apéndice B.
  - textInput: campo de texto en el cual el usuario puede introducir datos mediante entrada de teclado. Ejemplificado en las cajas de la segunda fila de figura B.5 en el apéndice B.
  - selectInput: ventana desplegable con valores predeterminados mediante el cual el usuario puede seleccionar uno de los valores. En la referenciada anteriormente también se puede ver este elemento.
  - checkboxGroupInput: elemento de selección múltiple.
  - numericInput: campo de texto en el que el usuario puede introducir un valor numérico.
  - Shinydirbutton: botón específico del paquete shinyFiles que permite la selección de directorios o carpetas, así como la creación de nuevos directorios. Ejemplificado en la figura B.2.
  - fileInput: elemento de selección de archivos, mediante el cual se cargan archivos de forma temporal en la aplicación para su uso. Permite determinar los tipos y tamaños de dichos ficheros. Ejemplificado en la figura B.4 para la subida del fichero FASTA.

La interfaz tiene partes dinámicas que se generan a medida que el usuario introduce los datos, ya que dependen directamente de los valores determinados mediante algunos *widgets*. Para las parte dinámicas de la interfaz se han utilizado los elementos que se listan a continuación. Estos elementos son simples indicadores de partes de la interfaz, gráficas y tablas, los cuales toman valores a través de las funciones del servidor, *render*, que se muestran en al figura 6.2.

- uiOutput: función encargada de actualizar valores dinámicos de la interfaz a partir de valores de entrada. Este elemento se ha utilizado en las pantallas que dependan del número variable de condiciones experimentales. La creación de *tabBoxes* con las condiciones experimentales o las réplicas de cada experimento son un ejemplo de ello.
- tableOutput: elemento de generación de tablas de manera dinámica. Se ha utilizado para mostrar los datos asociados al análisis diferencial y conteo de los genes que se han generado mediante los programas. Estos datos mostrados entabla son los que se utilizarán para la generación de gráficos.
- plotOutput: esta función de Shiny nos permite actualizar y mostrar elementos gráficos.

Además de estos elementos, que son los que dotan de funcionalidad y dinamicidad a la interfaz hemos de mencionar que la organización de los *widgets* y elementos dinámicos se ha realizado mediante los *box* y los *tabBox* que proporciona shinydashboard.

- Los *box* se han utilizado para agrupar elementos.
- Los *tabBox* se han utilizado para agrupar los resultados de cada condición experimental en una pestaña.

6.4.2. server.R

El servidor recoge la funcionalidad del controlador de la aplicación. En él se han implementado las funciones necesarias para actualizar la interfaz a medida que el usuario introduce los datos requeridos. Nos encontramos dos tipos de funciones: las encargadas de la actualización de la GUI, *render*, y las funciones reactivas a eventos, *observeEvent*.

Las funciones *render* se ejecutan de forma automática ante el cambio de los *widgets* y otras variables, que normalmente modifican sus valores a partir de las entradas del usuario. Por otra parte, las funciones *observeEvent* se ejecutan ante la activación de botones y se encargan de recoger la información que hasta ese momento haya proporcionado el usuario e de invocar la acción o la herramienta asociada al botón pulsado. Con estas funcionalidades podemos diferenciar dos grandes partes: la generación dinámica de elementos de la interfaz y la invocación de las herramientas y procesamiento los datos generados.

Funciones *render*

Para la generación y actualización de las partes dinámicas de la interfaz se han utilizado las siguientes funciones:

- *renderUI*: actualización o creación de elementos como *widgets*, *box* o *tabBox*.
- *renderPlot*: actualización o creación de gráficas con los resultados del análisis.

- *renderTable* o *renderDataTale*: actualización o creación de tablas y los datos de las mismas.

Las funciones son reactivas a uno o varios valores de entrada. A partir de estos valores se generan los elementos necesarios de entre los explicados en la sección 6.4.1 en el *Body*. De esta forma hemos conseguido generar un número variable de elementos con identificadores que nos permitirán recoger e introducir todos los valores necesarios asociados a las condiciones experimentales y las réplicas que el usuario quiera proporcionar.

Es importante mencionar que si se modifican los valores de entrada, las funciones se desencadenarán automáticamente y los elementos dinámicos asociados a estas funciones se perderán. Se generarán los nuevos elementos y los datos asociados a los elementos antiguos habrán de ser introducidos de nuevo o cambiarlos por los nuevos valores.

En la mayoría de los casos, las funciones *renderUI* son reactivas a las variables asociadas a los *widgets*. Por el contrario, las funciones *renderTable* y *renderPlot* tiene como valores de entrada las variables generadas en las funciones *observeEvent*, ya que se generarán las tablas y gráficas con los resultados generados tras la ejecución de las herramientas del análisis.

Función *observeEvent*

Por otra parte, la invocación de las herramientas se controla mediante la interacción con botones de la interfaz gráfica. Para ello se ha utilizado la función *observEvent*. Esta es una función que se activa mediante el clic en dichos botones. Una vez se ha presionado el botón asociado a un evento se procede a hacer una recogida de los datos de entrada, la invocación a la herramienta (o acción) asociada y procesamiento de los resultados obtenidos. Los eventos que se han implementado con esta funcionalidad son:

EV.1 **Comenzar**: botón asociado a la pantalla de la figura G.1 de la maqueta **M.1**.
Se encarga de tomar el valor determinado mediante shinyDirButton y de crear en dicho directorio el sistema de carpetas en los que se almacenaran los datos y resultados generados en cada uno de los pasos del análisis. Además, se recogen los *paths* a las herramientas del *workflow*. Por último esta acción se encarga de proporcionar los premisos de ejecución a los *scripts* y ejecutables de las herramientas que lo precisen.

EV.2 **Crear índices**: botón asociado a la pantalla de la figura G.2 de la maqueta **M.3**.
Toma el fichero de datos del genoma de referencia así como el directorio en el que se ha determinado la localización de la herramienta Bowtie y a partir de estos datos crea el comando de ejecución de la herramienta, almacenando los resultados en el directorio correspondiente.

EV.3 **Ejecutar Bowtie**: botón asociado a la pantalla de la figura G.3 de la maqueta **M.4**.
Este es el control más complejo, en el cual se han de recopilar la gran mayoría de los datos proporcionados por el usuario. Se han de recoger los parámetros de ejecución del Bowtie, los datos asociados a la condición de control así como de todas las réplicas proporcionadas para cada una de las condiciones a analizar. Con esto nos referimos a los nombres de las condiciones, el número de réplicas y los modos de alineación de las alineaciones. Con estos datos se hacen llamadas individuales a la herramienta Bowtie para cada réplica proporcionada, con las que se obtienen los ficheros de alineación y los ficheros con las estadísticas generadas por la herramienta.

A partir de este momento se trabajará con los datos generados por la aplicación evitando así que el usuario tenga que encargarse de gestionar los datos que han de utilizar el resto de las herramientas.

EV.4 Ejecutar Samtools: botón asociado a la pantalla de la figura G.5 de la maqueta **M.6**.
Para ordenar e indexar las alineaciones se recogen los ficheros de alineación generados por Bowtie y los parámetros de la herramienta Samtools proporcionados mediante la interfaz. A partir de estos datos se invoca a la herramienta y se almacenan los resultados en la carpeta de alineaciones del directorio de trabajo. Se consiguen los datos ordenados y en formato SAM y BAM.

EV.5 Ejecutar el contador y HTSeq: botón asociado a la pantalla de la figura G.6 de las maquetas **M.8** y **M.9**.
Para la ejecución del contador desarrollado en este proyecto se necesitan obtener los parámetros de ejecución de la herramienta, determinar si una réplica es *paired-end* o *single-end*, que tipo de característica se quiere contar y los separadores a utilizar. A partir de ellos se generan las llamadas pertinentes, correspondiéndose estas una por cada condición de análisis frente a la condición de control. Este análisis almacena los datos de la matriz de conteo en el directorio contador. En el caso de HTSeq se realiza el conteo de la condición de control una única vez.

EV.6 Análisis con DESeq: botones asociados a la pantalla de la figura G.6 de la maqueta **M.11**.
Se proporcionan tantos botones de ejecución como condiciones experimentales no de control como se hayan proporcionado. Mediante la pulsación de cada uno de ellos se produce la ejecución del código incluido en la sección D.3 del apéndice D. Dependiendo de qué condición experimental se quiere analizar se toman los datos de la matriz de conteo correspondiente a dicha condición frente a la condición de control. Una vez realizado el análisis se muestran los resultados mediante distintas gráficas y tablas.

Para conseguir todo lo anterior se han utilizado funciones para la recogida de textos, valores numéricos, los *path* a los directorios con las herramientas a ejecutar y el directorio de trabajo en el que almacenar los datos. A partir de esos datos se construyen los comandos de ejecución. Los textos de los comandos se ejecutan mediante la llamada a *system* y los resultados se almacenan en el directorio de trabajo. Una vez generados, se recogen los resultados y se utilizan para la representación gráfica.

Puesto que la mayoría de estas ejecuciones se realizan para cada una de las muestras proporcionadas, es necesario utilizar los bucles de tipo *lapply*, que permiten generar objetos en el caso de la renderización de la interfaz y actúan como un bucle *for* a la hora de realizar acciones. Para indicar al usuario que uno de estos bucles está siendo ejecutado se ha incluido un sistema de *progress bar* que indica en que momento de la ejecución se encuentra el sistema y permite determinar la finalización del mismo.

6.5. Modificaciones del diseño

Algunas de las maquetas diseñadas en el capítulo 5 han sufrido leves modificaciones tras la implementación. La funcionalidad ofrecida sigue satisfaciendo los requisitos definidos en el capítulo 4. El resultado visual de la interfaz se puede ver en las imágenes del apéndice B.

7 Pruebas

7.1. Bases de pruebas

Para verificar y validar la interfaz en su conjunto y las herramientas desarrolladas, así como la invocación a herramientas de análisis RNA-seq existentes, se han llevado a cabo diferentes tipos de pruebas: inspección de código, pruebas unitarias de caja negra y pruebas sobre la interfaz de usuario. Dado el carácter del trabajo las pruebas sobre las que se ha invertido mas tiempo es en la tercera, sobre la interfaz de usuario y las herramientas desarrolladas en el trabajo, es decir, el contador de genes y el motor de gráficas. En el caso de las herramientas utilizadas y desarrolladas por terceros, se asume su correcto funcionamiento y su mantenimiento por parte de los creadores de las mismas.

Además se han realizado pruebas sobre el correcto análisis de los datos para lo que se han analizado datos simulados y datos reales de los que se disponía de los resultados del análisis.

7.1.1. Inspección del código

La inspección de código ha sido constante a lo largo del tiempo debido a la metodología en cascada (figura 3.1) adoptada para el desarrollo del proyecto. Las diferentes fases de implementación han servido para mejorar la calidad del código a medida que el proyecto evolucionaba. Lo que se ha conseguido con este proceso es:

- Corregir de errores de implementación y funcionalidad.
- Aumentar del control de errores proporcionando mayor robustez.
- Mejorar la estructura del proyecto.

7.1.2. Pruebas unitarias de caja negra

Las pruebas unitarias llevadas a cabo han sido de tipo caja negra. Éstas se caracterizan por verificar que las salidas de cada una de las partes de la aplicación son las esperadas para las

entradas probadas. Se ha probado cada una de las fases del análisis de expresión diferencial integrado en la interfaz, utilizando como entrada tanto datos de RNA-seq simulados mediante la herramienta Polyester [38] como parte de los datos de secuenciación reales analizados en [39].

La ventaja de utilizar datos simulados es que se conocía la salida esperada de la expresión diferencial, puesto que una de las funcionalidades de la herramienta Polyester es simular lecturas RNA-seq en base a una matriz de *fold-changes* dada. Para probar todos los escenarios posibles, se simularon tanto lecturas *paired-end* como *single-end* para tres condiciones experimentales más la condición de control y se generaron tres réplicas por condición. También se modificaron parámetros relativos a las calidades de las lecturas para comprobar el correcto funcionamiento de las gráficas asociadas al alineador Bowtie.

Por otra parte, la verificación con datos RNA-seq obtenidos a partir de experimentos reales permitió garantizar el correcto funcionamiento del *pipeline* en comparación con las técnicas utilizadas actualmente para el análisis de datos RNA-seq en células procariotas.

Las más importantes son:

- Pruebas de ejecución de Bowtie
  Se ha comprobado que la generación de comandos UNIX para la ejecución de la herramienta contiene todos los campos necesarios y que se generan tantos como réplicas se han proporcionado. Se ha probado Bowtie para la generación de índices y generación de alineaciones. Para probar el Bowtie se han generado ficheros FASTA y FASTQ con pocos datos para entender su funcionalidad y verificar la correcta invocación del código.

- Pruebas de ejecución del Contador del TFG
  Se ha comprobado que los métodos devuelven los errores adecuados y que la generación de resultados es correcta. Para estas pruebas ha sido especialmente útil el uso de los datos simulados puesto que era posible conocer la matriz de conteo esperada y así contrastarla con el resultado del contador.

- Pruebas de ejecución del HTSeq
  Se ha comprobado la generación de comandos UNIX para la ejecución de la herramienta, que contiene los campos determinados por los usuarios. Para comprobar la correcta ejecución se han comprobado los ficheros de datos generados. Además, se ha probado que se puede generar la matriz de conteo resultante de la unión de los ficheros de conteo.

- Pruebas de ejecución del DESeq
  Se han probado las funciones utilizadas del paquete DESeq de *Bioconductor*. Además, se ha verificado la corrección del análisis implementado a partir de dichas funciones mediante la comparación con los resultados de los datos simulado y los datos reales sobre los que se ha realizado el análisis.

7.1.3. Pruebas sobre la interfaz de usuario

La interfaz de usuario se ha probado tanto para verificar el correcto funcionamiento de la aplicación como para validarla. Las pruebas se han llevado a cabo en un entorno UNIX con sistema operativo LINUX Ubuntu 14.4 y con una versión de R 3.2.2. Las versiones de las herramientas se detallan en el apéndice A, manual de instalación.

- Pruebas de la correcta navegación entre pantallas. Han consistido en ejecutar las acciones de navegación descritas en el anexo B, manual de utilización .

- Pruebas de cada pantalla de manera individual. Han demostrado el correcto funcionamiento de los campos dinámicos y la renderización de los mismos.
- Pruebas de la repercusión de las acciones realizadas en una pantalla sobre otras pantallas. Han verificado la funcionalidad de creación y actualización de los campos dinámicos.
- Pruebas de ejecución de tareas mediante botones. Se ha comprobado que se recogen correctamente los datos necesarios para la ejecución del análisis y que se construyen correctamente los comandos necesarios para su ejecución
- Pruebas de generación de resultados. Han demostrado que la generación de gráficas y tablas se realiza correctamente.
- Pruebas de verificación de las gráficas mediante la comparativa de los resultados generados por las herramientas y los datos representados en las gráficas.

7.1.4. Validación

La validación de la aplicación consiste en constatar que satisface el propósito para el que fue planteada. Se comprueba, por tanto, que cumpla los requisitos especificados en el capítulo 4, tanto los funcionales como los no funcionales.

En este contexto, se ha confirmado que existe una forma de satisfacer, mediante la interfaz gráfica, cada uno de los requisito funcionales, se ha validado su cumplimiento. Además se han comprobado aquellos requisitos funcionales asociados a las herramientas del *pipeline* del análisis.

Con este propósito se ha determinado, mediante un formulario de respuesta múltiple, con que facilidad los usuarios pueden completar cada una de las partes del análisis ofrecidas en la interfaz. Este formulario se muestra en la tabla 7.1. Han colaborado para ello la tutora, Irene Rodríguez, y potenciales futuros usuarios.

La facilidad o dificultad con la que se completa cada fase del análisis se valora numéricamente como sigue:

0. No se ha conseguido completar el análisis.
1. Se ha completado la tarea, pero se ha necesitado ayuda externa por parte del autor.
2. Se ha completado la tarea, pero se ha necesitado tiempo para conseguir entender los pasos a seguir.
3. Se ha completado la tarea, a partir de la información proporcionada a modo de explicación en la aplicación.
4. Se ha completado la tarea, de una forma intuitiva y fluida.
5. No se ha encontrado ninguna dificultad a la hora de realizar el análisis.

7.2. Resultados

Las distintas pruebas de verificación han tenido las siguientes consecuencias:

Tarea	Valoracion	Comentarios
Seleccionar las herramientas y el directorio de trabajo para realizar el análisis		
Crear los índices del genoma de referencia		
Realizar las alineaciones con la herramienta Bowtie		
Ordenar e indexar las alineaciones con la herramienta Samtools		
Realizar el conteo de genes con la herramienta desarrollada en el TFG		
Realizar el conteo de genes con la herramienta HTSeq		
Ejecutar el análisis de expresión diferencial mediante la herramienta HTSeq		
Obtener y visualizar los resultados por medio de las gráficas y tablas		

Tabla 7.1: Tareas que hay que validar y valorar

- Se ha refactorizado parte de la implementación, mejorando su estructura interna, claridad y mantenibilidad, fruto de las inspecciones del código.
- Se han corregido errores de ejecución de las herramientas y errores leves gracias a las pruebas de caja negra.
- Se han podido detectar y corregir errores de la interfaz gráfica, resultado de las pruebas realizadas sobre la aplicación. Esto ha llevado a mejorar el control de errores y mensajes al usuario.
- Se han realizado modificaciones sobre la interfaz con el fin de mejorar aquellos aspectos que presentaban la mayor complejidad.

Respecto a las pruebas de validación, el formulario ha permitido comprobar como de fácil o difícil es manejar la interfaz y conseguir un análisis satisfactorio. Los resultados son aceptables, pero se ha de aumentar la muestra de usuarios que lleven a cabo este test de validación. En todo caso, el manual de usuario proporcionado en el anexo B proporciona la información necesaria para la utilización de la herramienta.

Se ha comprobado que las funciones implementadas generan los resultados correctamente, mediante los gráficos. En el caso de los datos simulados se ha comprobado el correcto resultado a partir de la matriz de simulación, comprobando que los genes diferencialmente expresados coinciden con los que se habían especificado al generar los datos, teniendo en cuenta el FDR que se ha especificado como parámetro.

Respecto a los datos reales y de acuerdo a los resultados reflejados en [39] y en el material suplementario asociado a este artículo, los resultados de análisis de expresión diferencial obtenidos con nuestra aplicación coincidían casi en su totalidad con los resultados reportados en [39], aunque no se esperaban coincidencias exactas dado que el *pipeline* adoptada en ambos casos difiere.

8

Conclusiones y trabajo futuro

8.1. Conclusiones

El presente Trabajo de Fin de Grado ha permitido el desarrollo de una aplicación web destinada a integrar un *workflow* para el análisis de expresión diferencial permitiendo un uso sencillo e intuitivo de las herramientas recogidas en el mismo. Ha supuesto aprender las bases de la bioinformática relacionada con la genómica (ver apéndice H), en particular lo referente a la expresión diferencial de genes, conocer las herramientas utilizadas e investigar las soluciones existentes hasta el momento con el fin de encontrar sus limitaciones. Este trabajo se ha realizado con el fin de poder crear una interfaz que utilice un *pipeline* unificado y que a la vez simplifique el uso de estas herramientas y lectura de los resultados al usuario no experto.

El proyecto ha seguido un ciclo de vida en cascada con retroalimentación en el que se han seguido las fases de análisis de requisitos, diseño, implementación, pruebas y validación hasta llegar a un producto que satisface la idea inicial del proyecto, pero que a su vez ha ido evolucionando a medida que crecía el conocimiento del problema. Actualmente se encuentra en fase mantenimiento y ampliación de funcionalidad.

El proyecto se ha diseñado siguiendo un patrón arquitectura MVC (modelo-vista-controlador) y se ha desarrollado a partir de una mezcla de lenguajes de programación, siendo estos principalmente R y C. El modelo se ha basado principalmente en la utilización de C para la implementación del contador y herramientas auxiliares, e indirectamente los lenguajes de programación que utilizan las herramientas integradas en la aplicación como son Python para HTSeq y R para DESeq. El modelo integra las herramientas seleccionadas para el *pipeline*. Para la vista y el controlador se ha utilizado R, en particular las librerías de Shiny (*shinydashboard, shinyFiles*). El modelo de datos ha integrado algunos de los formatos de ficheros comúnmente utilizados en el análisis de datos RNA-seq como son los formatos FASTA, FASTQ, GFF, SAM. Además, se han utilizado ficheros de texto para el almacenamiento de las matrices de conteo de expresión génica y resultados de los análisis. Además, se han utilizado ficheros de texto para el almacenamiento de las matrices de conteo de expresión génica y resultados de los análisis. Los resultados se han hecho visibles al usuario final mediante el uso de la librería *ggplot2* de R, la cual ha sido utilizada para integrar en la interfaz los distintos gráficos asociados a cada una de las fases y

herramientas del análisis.

Durante el transcurso del Trabajo de Fin de Grado se han podido poner en práctica los conocimientos adquiridos durante la carrera, sobre todo los conocimientos adquiridos en el Grado de Ingeniería Informática. *Ingeniería del Software* ha aportado los conocimientos sobre cómo gestionar un proyecto de software y *Sistemas Informáticos I y II* ha aportado los conocimientos necesarios para el desarrollo de una aplicación web siguiendo el patrón de arquitectura MVC (modelo-vista-controlador). Las asignaturas del Grado en Matemáticas como *Estadística* y *Probabilidad* han aportado la base necesaria para entender las estimaciones del análisis de expresión diferencial. La base de conocimientos y metodología de aprendizaje rápida y eficaz adquiridas durante el Doble Grado de Ingeniería Informática y Matemáticas han permitido la consecución del objetivo de este proyecto, cuyo desarrollo requería el aprendizaje y uso extensivo del lenguaje de programación R, no estudiado hasta el momento.

En resumen, la experiencia de la elaboración del Trabajo de Fin de Grado ha sido muy enriquecedora y gratificante, suponiendo una ampliación de conocimientos tanto a nivel informático como a nivel multidisciplinar, en este caso en el campo de la bioinformática. Por una parte se ha construido desde cero un proyecto de envergadura considerable. El paso por sus distintas fases y la visión de su evolución ha sido muy valioso a la hora de conocer la realidad del desarrollo de un proyecto propio. Durante la implementación se ha aprendido a utilizar R, centrando los esfuerzos en el desarrollo de la interfaz gráfica y el controlador de la aplicación. Además se ha adquirido el conocimiento necesario para la integración de herramientas en una aplicación web y la utilización de datos a la hora de representar resultados en este lenguaje de programación. Por otra parte se ha mejorado el uso de las herramientas utilizadas durante el desarrollo de la aplicación como son: RStudio [25], Sublime Text [31], LaTeX[33], Cacoo [32].

8.2. Líneas de trabajo futuro

Este Trabajo de Fin de Grado tenía como finalidad inicial el desarrollo de una interfaz gráfica que permitiese realizar un análisis de expresión diferencial a usuarios no expertos en informática y/o bioinformática. Una vez conseguido el objetivo inicial, se han incluido en el proyecto funciones adicionales que complementan la aplicación. No obstante, el proyecto está sujeto a una serie de mejoras que se describen a continuación.

Añadir nuevas herramientas al *pipeline*

Esta mejora consistiría en añadir módulos de funcionalidad a la aplicación, totalmente independientes de la aplicación actual, ya que la misma es fácilmente escalable en esta dirección. El trabajo consistiría en añadir la parte de la interfaz gráfica asociada a la nueva herramienta y el controlador de la misma. Una de las fases del análisis a añadir sería el control de calidades realizado antes de las alineaciones (paso 2 de la figura F.2). El modelo de datos también escalable ya que cada una de las herramientas trabaja con un tipo de datos estándar.

Trabajo con células eucariotas

Por el momento sólo se proporciona la funcionalidad necesaria para reproducir un análisis sobre células de tipo procariota. Sería necesario realizar un estudio de las cualidades necesarias

para poder realizar un análisis de expresión diferencial sobre las células eucariotas que nos permita encontrar las diferencias en el *pipeline* y la funcionalidad a añadir.

Permitir análisis de expresión diferencial para múltiples condiciones experimentales simultáneas

Actualmente el análisis de expresión diferencial se realiza contrastando una condición experimental frente a una condición de control. Una de las mejoras que se proponen es la realización de un análisis de varias condiciones experimentales simultáneamente, es decir, encontrar aquellos genes diferencialmente expresados en una de las condiciones experimentales frente al resto. Para llevar a cabo esta mejora, sería necesario profundizar en el estudio de las herramientas como DESeq y sus funciones para realizar el análisis múltiple.

Publicar la interfaz desarrollada para su uso público

La publicación de la herramienta en un repositorio como GitHub [40] permitiría que una gran cantidad de usuarios tuvieran acceso a la aplicación, lo cual es altamente deseable dado el propósito eminentemente práctico de este proyecto. Además, las opiniones y sugerencias de los usuarios serían una buena fuente de información para el mantenimiento, control de errores y mejora de la aplicación.

Almacenamiento de la información

Mediante esta mejora se podría almacenar el estado de la aplicación, lo que permitirá reanudar un análisis de expresión diferencial a partir del estado en el que se dejó. Una posibilidad es, al cerrar la aplicación, crear un archivo que contenga los datos de cada una de las variables de usuario. Una vez que se abra de nuevo la aplicación se buscaría este archivo en el directorio de trabajo y en caso de existir se establecerían los valores por defecto con los valores almacenados en el archivo de *backup*.

Bibliografía

[1] SAM/BAM Format Specification Working Group et al. Sequence alignment/map format specification. *Available at ht tps://github. com/samtools/hts-specs*, 2013.

[2] Nuno A Fonseca, Johan Rung, Alvis Brazma, and John C Marioni. Tools for mapping high-throughput sequencing data. *Bioinformatics*, page bts605, 2012.

[3] Samtools tutorial. `http://biobits.org/samtools_primer.html`. Accessed: 2015-11-22.

[4] Simon Anders, Davis J McCarthy, Yunshun Chen, Michal Okoniewski, Gordon K Smyth, Wolfgang Huber, and Mark D Robinson. Count-based differential expression analysis of rna sequencing data using r and bioconductor. *Nature protocols*, 8(9):1765–1786, 2013.

[5] htseq-count. `http://www-huber.embl.de/users/anders/HTSeq/doc/count.html`. Accessed: 2015-12-22.

[6] Bioinformatics. `http://www.pasteur.fr/recherche/unites/Binfs/definition/bioinformatics_definition.html`. Accessed: 2015-09-25.

[7] Dna sequencing. `http://www.ncbi.nlm.nih.gov/books/NBK21117/`. Accessed: 2015-09-25.

[8] Martin A Perdacher. Next-generation sequencing and its applications in rna-seq.

[9] Zhong Wang, Mark Gerstein, and Michael Snyder. Rna-seq: a revolutionary tool for transcriptomics. *Nature Reviews. Genetics*, 10(1):57–63, 2009.

[10] Gilbert SF. Differential gene expression. `http://www.ncbi.nlm.nih.gov/books/NBK10061/`. Accessed: 2015-09-25.

[11] Vanessa M Kvam, Peng Liu, and Yaqing Si. A comparison of statistical methods for detecting differentially expressed genes from rna-seq data. *American journal of botany*, 99(2):248–256, 2012.

[12] William Michael Landau and Peng Liu. Dispersion estimation and its effect on test performance in rna-seq data analysis: a simulation-based comparison of methods. 2013.

[13] Ségolène Caboche, Christophe Audebert, Yves Lemoine, and David Hot. Comparison of mapping algorithms used in high-throughput sequencing: application to ion torrent data. *BMC genomics*, 15(1):264, 2014.

[14] Charlotte Soneson and Mauro Delorenzi. A comparison of methods for differential expression analysis of rna-seq data. *BMC bioinformatics*, 14(1):91, 2013.

[15] Illumina. `http://www.illumina.com/takeover.html`. Accessed: 2015-09-25.

[16] Hts mappers. `http://www.ebi.ac.uk/~nf/hts_mappers/`. Accessed: 2015-12-22.

[17] Prokaryotes. https://en.wikipedia.org/wiki/Prokaryote. Accessed: 2015-09-25.

[18] Simon Anders, Paul Theodor Pyl, and Wolfgang Huber. Htseq–a python framework to work with high-throughput sequencing data. *Bioinformatics*, page btu638, 2014.

[19] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015.

[20] Simon Anders and Wolfgang Huber. Differential expression analysis for sequence count data. *Genome Biology*, 11:R106, 2010.

[21] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 289–300, 1995.

[22] Galaxy. https://usegalaxy.org/. Accessed: 2015-12-16.

[23] Rockhopper. http://cs.wellesley.edu/~btjaden/Rockhopper/. Accessed: 2015-12-15.

[24] Rnaseqgui. http://bioinfo.na.iac.cnr.it/RNASeqGUI/. Accessed: 2015-12-15.

[25] Rstudio. https://www.rstudio.com/. Accessed: 2015-12-16.

[26] Thomas Lin Pedersen. *shinyFiles: A Server-Side File System Viewer For Shiny*, 2015. R package version 0.6.0.

[27] Winston Chang. *shinydashboard: Create Dashboards with 'Shiny'*, 2015. R package version 0.5.1.

[28] Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie, and Jonathan McPherson. *shiny: Web Application Framework for R*, 2015. R package version 0.12.2.

[29] Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009.

[30] Henrik Bengtsson. *matrixStats: Functions that Apply to Rows and Columns of Matrices (and to Vectors)*, 2015. R package version 0.50.1.

[31] Sublime text. https://www.sublimetext.com/. Accessed: 2015-12-16.

[32] Cacoo. https://cacoo.com/. Accessed: 2015-12-16.

[33] Latex. http://www.latex-project.org/. Accessed: 2015-12-16.

[34] Shiny rstudio. http://shiny.rstudio.com/. Accessed: 2015-11-27.

[35] Shiny dashboard rstudio. https://rstudio.github.io/shinydashboard/. Accessed: 2015-11-27.

[36] Package shiny. https://cran.r-project.org/web/packages/shiny/shiny.pdf. Accessed: 2015-11-27.

[37] Package shiny dashboard. https://cran.r-project.org/web/packages/shinydashboard/shinydashboard.pdf. Accessed: 2015-11-27.

[38] Alyssa C Frazee, Andrew E Jaffe, Ben Langmead, and Jeffrey T Leek. Polyester: simulating rna-seq datasets with differential transcript expression. *Bioinformatics*, page btv272, 2015.

[39] Ryan McClure, Divya Balasubramanian, Yan Sun, Maksym Bobrovskyy, Paul Sumby, Caroline A Genco, Carin K Vanderpool, and Brian Tjaden. Computational analysis of bacterial rna-seq data. *Nucleic acids research*, 41(14):e140–e140, 2013.

[40] Github. `https://github.com/`. Accessed: 2016-01-11.

[41] Fasta. `http://blast.ncbi.nlm.nih.gov/blastcgihelp.shtml`. Accessed: 2015-11-13.

[42] Fastq. `http://maq.sourceforge.net/fastq.shtml`. Accessed: 2016-01-08.

[43] Coursera. `https://www.coursera.org/`. Accessed: 2015-09-27.

[44] Introduction to genomic technologies. `https://www.coursera.org/learn/genintro`. Accessed: 2015-09-27.

[45] Genomic data science with galaxy. `https://www.coursera.org/learn/gengalaxy`. Accessed: 2015-09-30.

[46] Python for genomic data science. `https://www.coursera.org/learn/genpython`. Accessed: 2015-10-07.

[47] Command line tools for genomic data science. `https://www.coursera.org/learn/gencommand`. Accessed: 2015-10-15.

[48] Statistics for genomic data science. `https://www.coursera.org/course/genstats`. Accessed: 2015-10-20.

Anexos

A

Manual de instalación

El proyecto se ha desarrollado y probado en una plataforma Linux con distribución Ubuntu(64-bit).

A.1. Listado de herramientas y versiones

Estas son las herramientas a utilizar con la aplicación.

- Bowtie versión *1.1.2*.
- Samtools versión *1.3*.
- HTSeq versión *0.6.1*.

Para trabajar con la herramienta se utilizará el IDE RStudio, con una version de R 3.2.2 (2015-08-14). Se deberán de descargar e instalar los siguientes paquetes:

- Shiny package versión *0.12.2*.
- Shinydashboard package versión *0.5.1*.
- ShinyFiles package versión *0.6.0*.
- DESeq package versión *1.20.0*.
- Ggplot2 package versión *2.0.0*.
- Reshape package versión *0.8.5*.
- MatrixStats package versión *0.50.1*.

Se puede ejecutar el *script* de R que realiza las acciones de descarga e instalación de dichas librerías

```
install.packages("shiny")
install.packages("shinydashboard")
install.packages("shinyFiles")
install.packages("ggplot2")
install.packages("reshape")
install.packages("matrixStats")
source("http://bioconductor.org/biocLite.R")
biocLite("DESeq")

library(shiny)
library(shinyFiles)
library(ggplot2)
library(reshape)
library(DESeq)
library(shinydashboard)
library(matrixStats)
```

B

Manual de utilización

El programa se lanza mediante la opción *Run app* que ofrece RStudio al abrir los ficheros asociados a una aplicación web, como son en este caso ui.R y server.R. Una vez lanzado el programa se navegará por las pantallas mediante el menú lateral. Las pantallas por las que podemos navegar y hacer el análisis se explican a continuación:

Start

En la pantalla B.1, que se muestra al seleccionar el item *Start* del menú lateral, se muestra al usuario una breve explicación del contenido de la aplicación. Se muestran los directorios en los que se guardarán los resultados del análisis y en los que se encuentran las herramientas a utilizar. Estos últimos se seleccionan mediante un sistema como el mostrado en la figura B.2. Una vez seleccionado el directorio de trabajo, se deberá dar comienzo al análisis mediante la creación de las carpetas donde se almacenarán los datos. Para ello se deberá pulsar el botón ***Start*** mostrado en la figura B.1. Este se corresponde con la funcionalidad explicada en **EV.1** de la sección 6.4.2. En caso de que no se haya seleccionado el directorio de alguna de las herramientas no será posible hacer uso de ella.

Bowtie

En la pantalla B.3, a la que se accede al seleccionar el item Bowtie del menú, se muestran los textos explicativos asociados a cada uno de los pasos del análisis realizado con la herramienta Bowtie. En ella se muestran los pasos a seguir en cada una de las siguientes fases del análisis, correspondiente a la creación de índices del genoma de referencia y a la alineación de los *reads*.

Create Index

La pantalla mostrada en la figura B.4, asociada al subitem *Create Index*, tiene como fin generar los índices sobre el genoma de referencia. Para ello el usuario deberá subir el fichero FASTA con el genoma de referencia y pulsar el botón ***Create Index***. Este se corresponde con la funcionalidad explicada en **EV.2** de la sección 6.4.2.

Align

La siguiente fase del análisis (ver figura B.5) es la encargada de recoger todos los datos proporcionados por el usuario. A ella se accede desde el subitem *Align* del menú. En primer lugar habrá de determinar el número de condiciones experimentales y el modo en las que querrá analizarlas con la herramienta Bowtie, especificando los modos de alineación y los parámetros de ejecución.

Una vez especificados estos parámetros el usuario deberá pasar a introducir los datos asociados a cada condición experimental y la condición de control. Estos datos se corresponden con el nombre de la condición experimental, el número de réplicas y para cada réplica los ficheros FASTQ y el tipo de lecturas proporcionadas.

En el caso de que se hayan seleccionado el tipo de lectura *single-end* sólo será necesario proporcionar un fichero FASTQ asociado a *forward* para la réplica, y en caso de seleccionar *paired-end* el usuario deberá proporcionar tanto los ficheros FASTQ *forward* como los *reverse*. Una vez proporcionados todos estos datos el usuario deberá pulsar el botón ***Run Bowtie*** lo cual generará los ficheros SAM con las alineaciones. Este se corresponde con la funcionalidad explicada en **EV.3** de la sección 6.4.2.

Results Bowtie

En esta pantalla se muestran los resultados obtenidos tras la ejecución de la herramienta Bowtie. Para acceder a ella hemos de seleccionar la opción de resultados dentro del item Bowtie del menú. En la pantalla B.6 se representan los datos estadísticos en tanto por ciento. En la pantalla B.7 se representan los datos numéricos y por último en la figura B.8 se muestran los datos asociados a la calidad de las alineaciones por posición de cada una de las condiciones experimentales.

Samtools

En la pantalla B.9 que obtenemos desde el menu latelar en la opcion llamada Samtools se presenta una combinación de texto explicativo de la misma y la herramienta Samtools. Para su uso el usuario deberá especificar el modo de ordenación que desee obtener. Una vez establecido deberá pulsar el botón ***Run Samtools***. Este se corresponde con la funcionalidad explicada en **EV.4** de la sección 6.4.2.

Counters

La figura B.10 contiene los textos explicativos referentes a los contadores.

TFG counter y HTSeq

Estas herramientas se acceden mediante los subitems denominados Counter TFG y HTSeq del menú Counters de la barra latera. Para proceder al conteo de genes se pueden elegir los contadores HTSeq o Contador. En la figura B.11 se expone el modo de conteo mediante el contador desarrollado en el TFG. Para utilizarlo el usuario ha de especificar los parámetros del solapamiento, la característica y el separador a utilizar. A continuación podrá subir el fichero de referencia de genes GFF, y una vez llevado a cabo se podrá ejecutar el conteo mediante la pulsación del botón ***Run Counter***. Este se corresponde con la funcionalidad explicada en **EV.5** de la sección 6.4.2.

La ejecución del conteo mediante HTSeq ser realiza igual que en el caso anterior, pero se han de especificar otros parámetros de ejecución adicionales.

Results counter

Los resultados del conteo de genes se muestran en forma de diagrama de barras tal y como muestra la figura B.13 y en la figura B.14. Se puede seleccionar el gen del que se quiere obtener la información y a partir de esta selección se muestran los resultados para la condición de control frente a la condición experimental de la que se está mostrando la información.

DESeq

Por último en la figura B.15, asociada al item DESeq del menú, proporciona la explicación para hacer uso del último paso del análisis. En este paso se realiza el análisis de expresión diferencial. Para ello se deben especificar los parámetros de ejecución del DESeq. Una vez determinados se debe pulsar el botón correspondiente a la condición experimental que se desea analizar. Una vez pulsado se generan los resultados en gráficos como se muestran en las figuras B.16, B.17 y B.18. Este se corresponde con la funcionalidad explicada en **EV.6** de la sección 6.4.2.

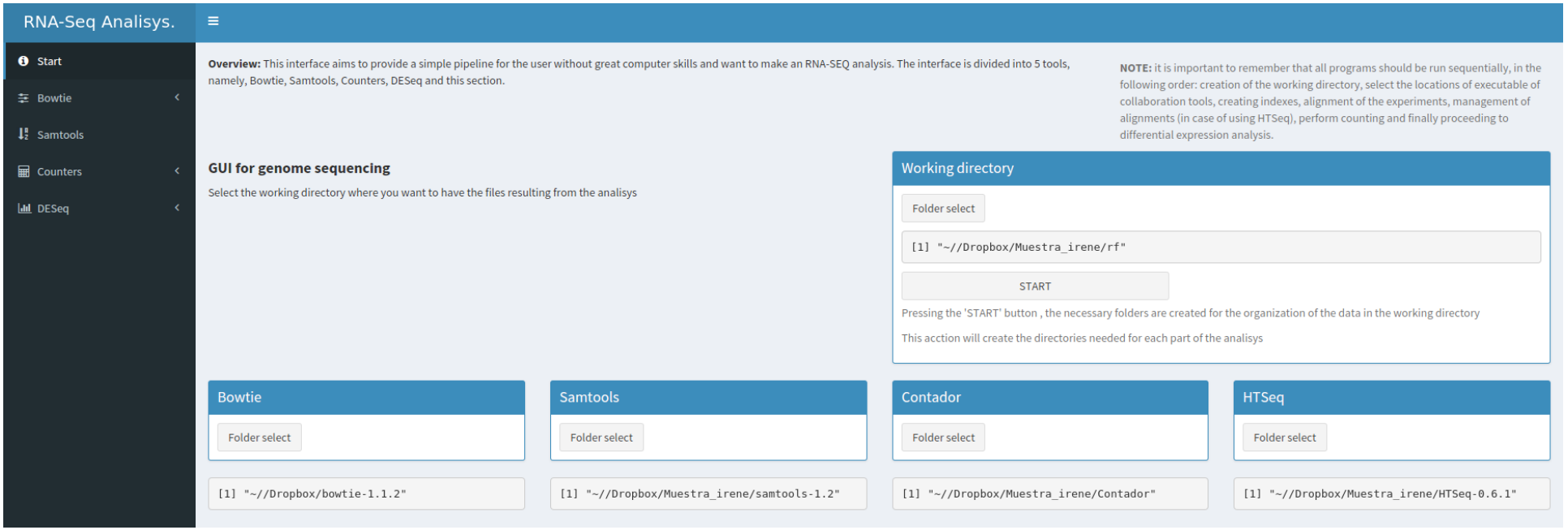


Figura B.1: Inicio y selección de directorios de trabajo

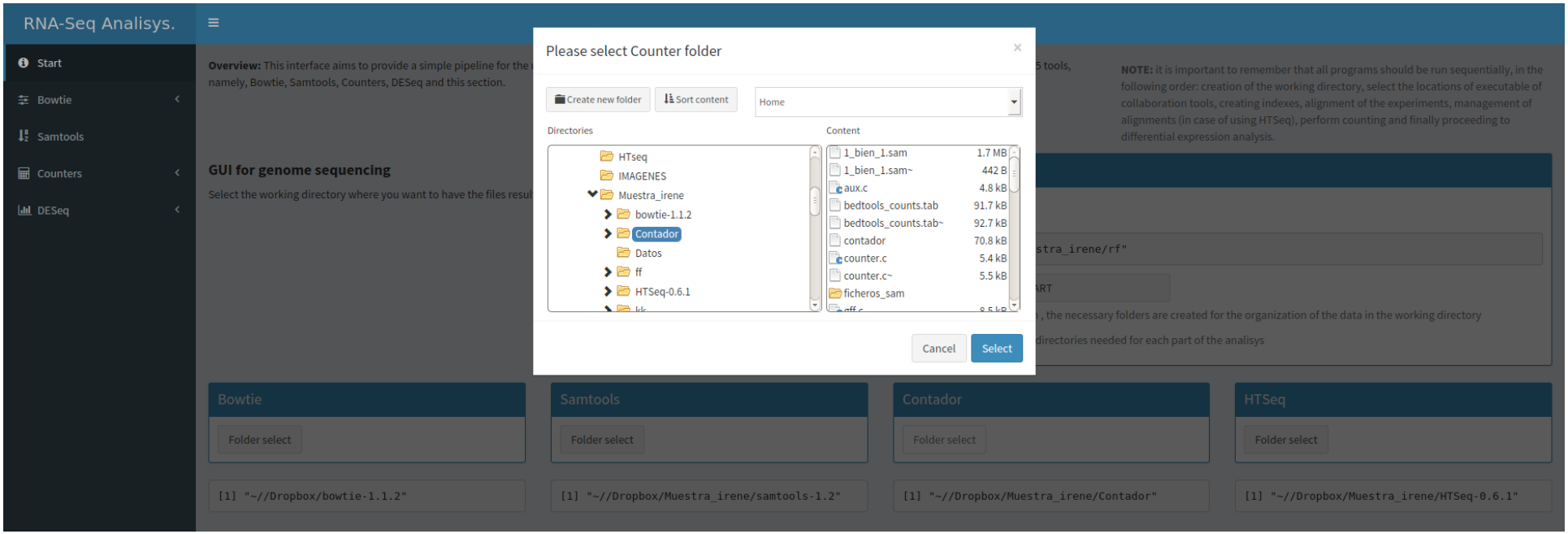


Figura B.2: Selección de directorios

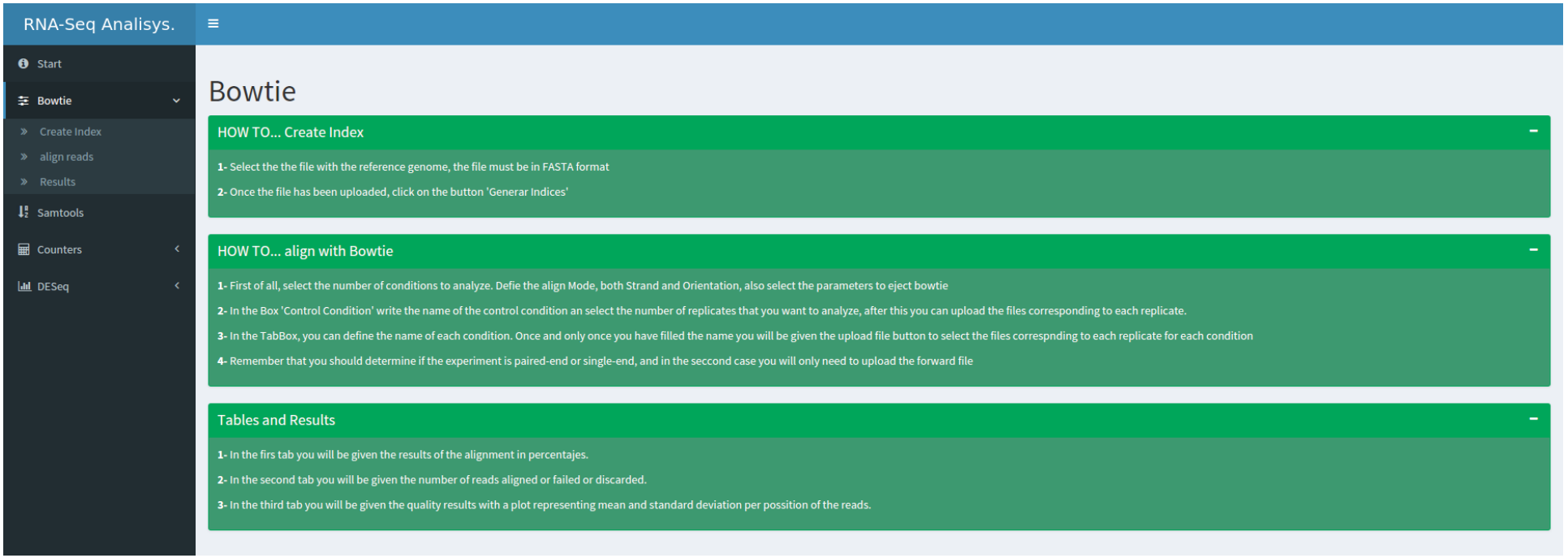


Figura B.3: Sección explicativa de la herramienta Bowtie

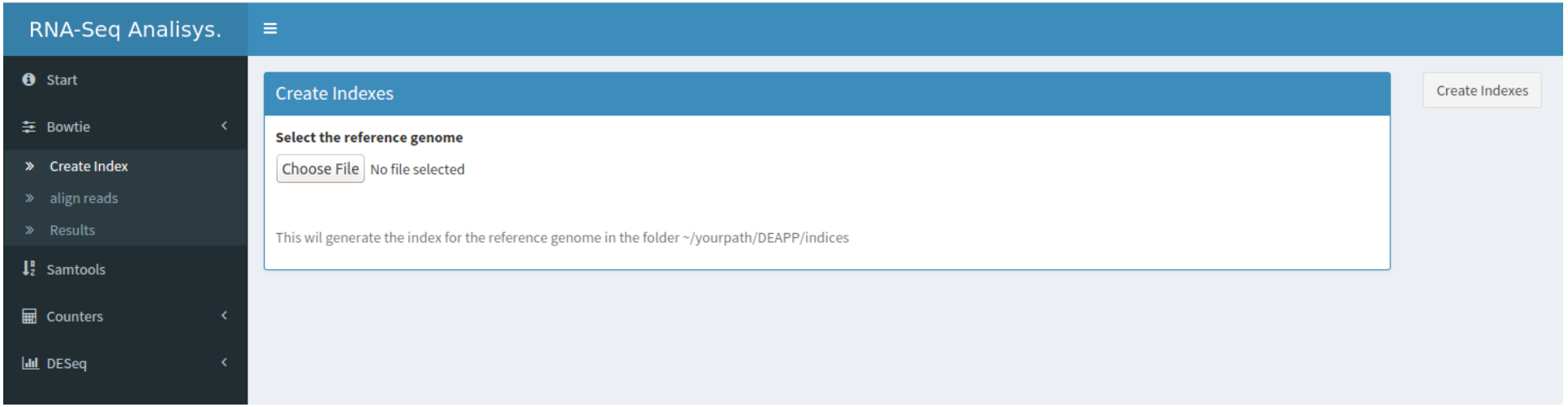


Figura B.4: Generación de índices sobre el genoma de referencia

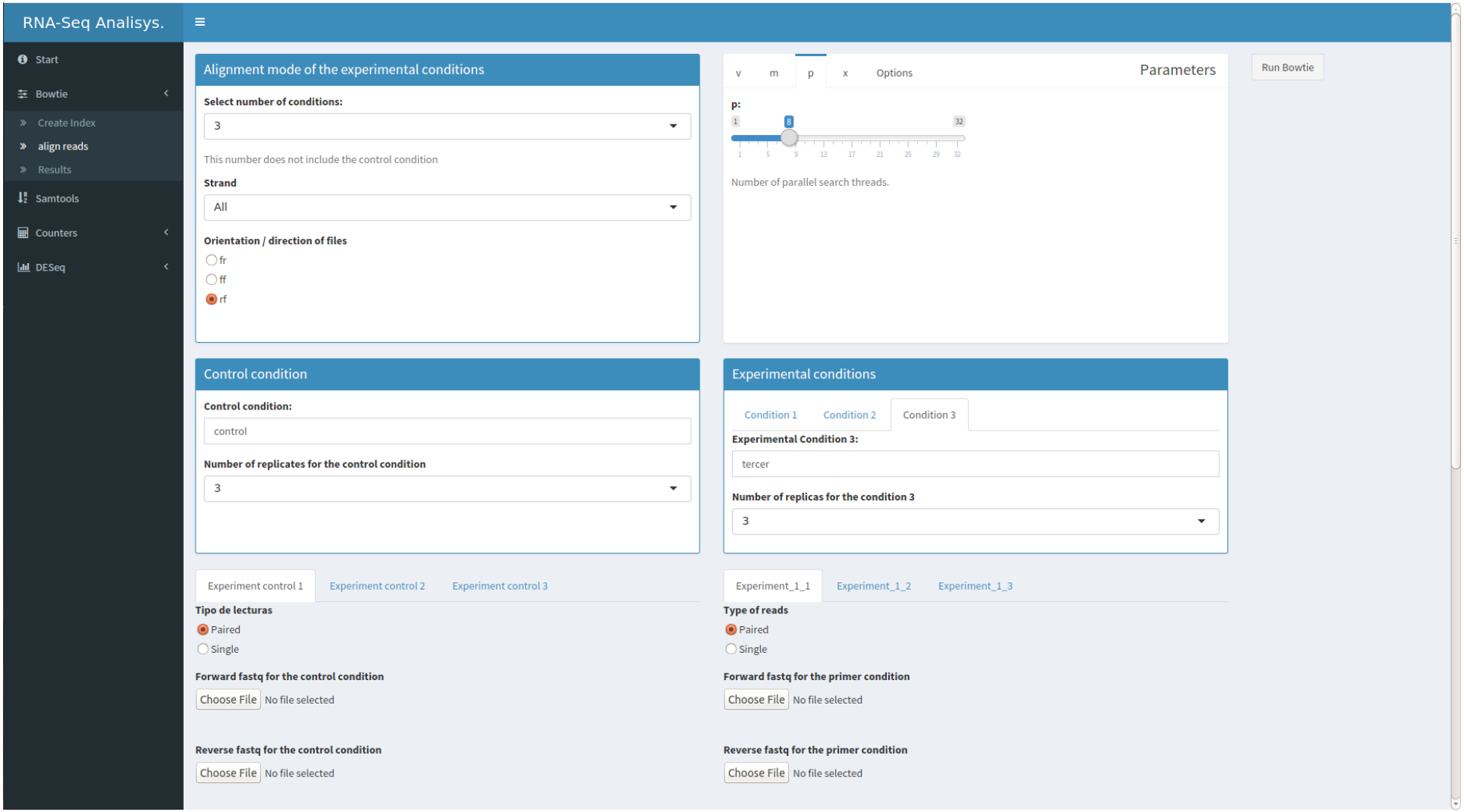


Figura B.5: Ejecución de Bowtie para la creación de alineaciones de las condiciones experimentales

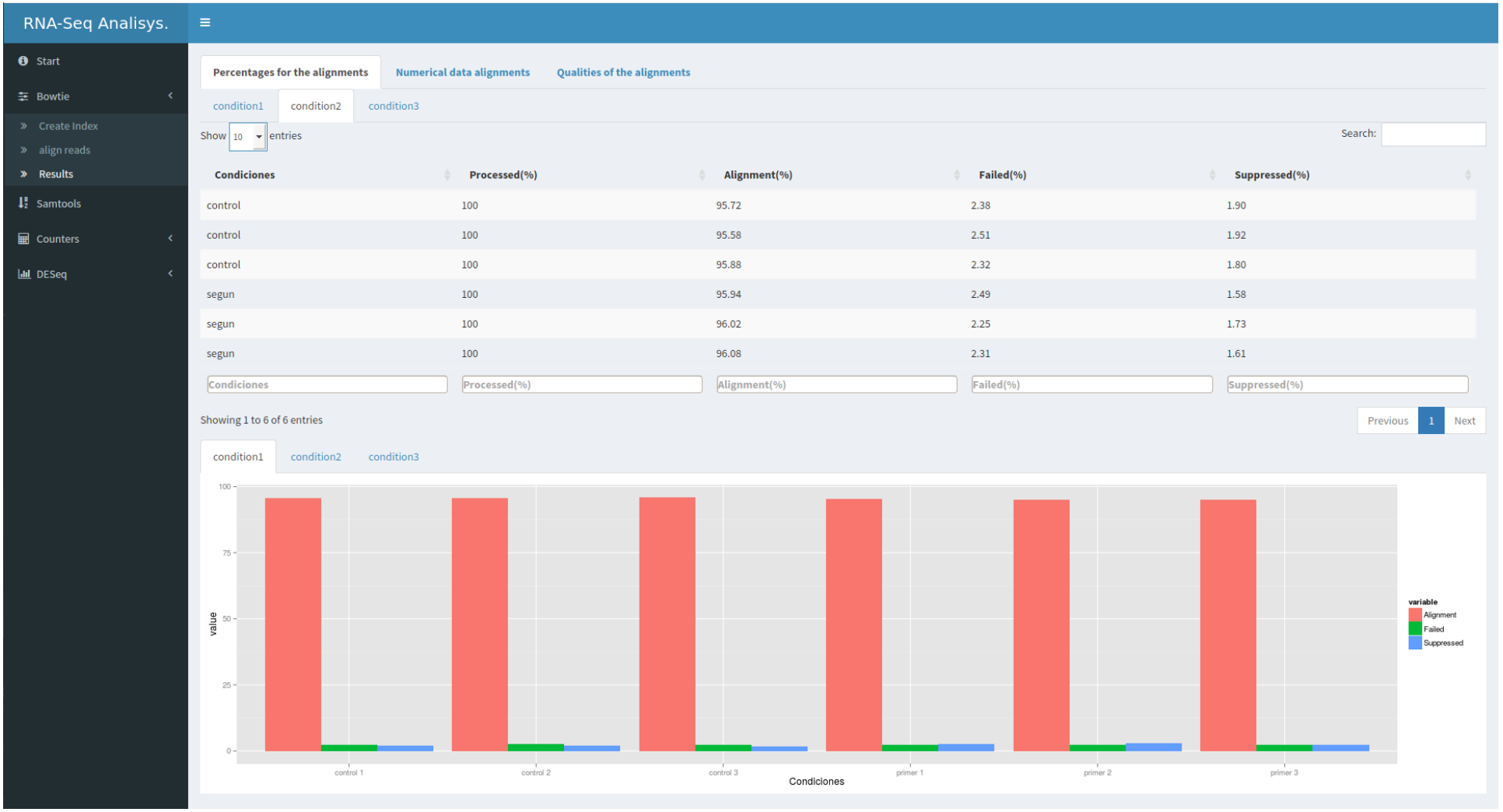


Figura B.6: Datos estadísticos de la ejecución de Bowtie en tanto por ciento

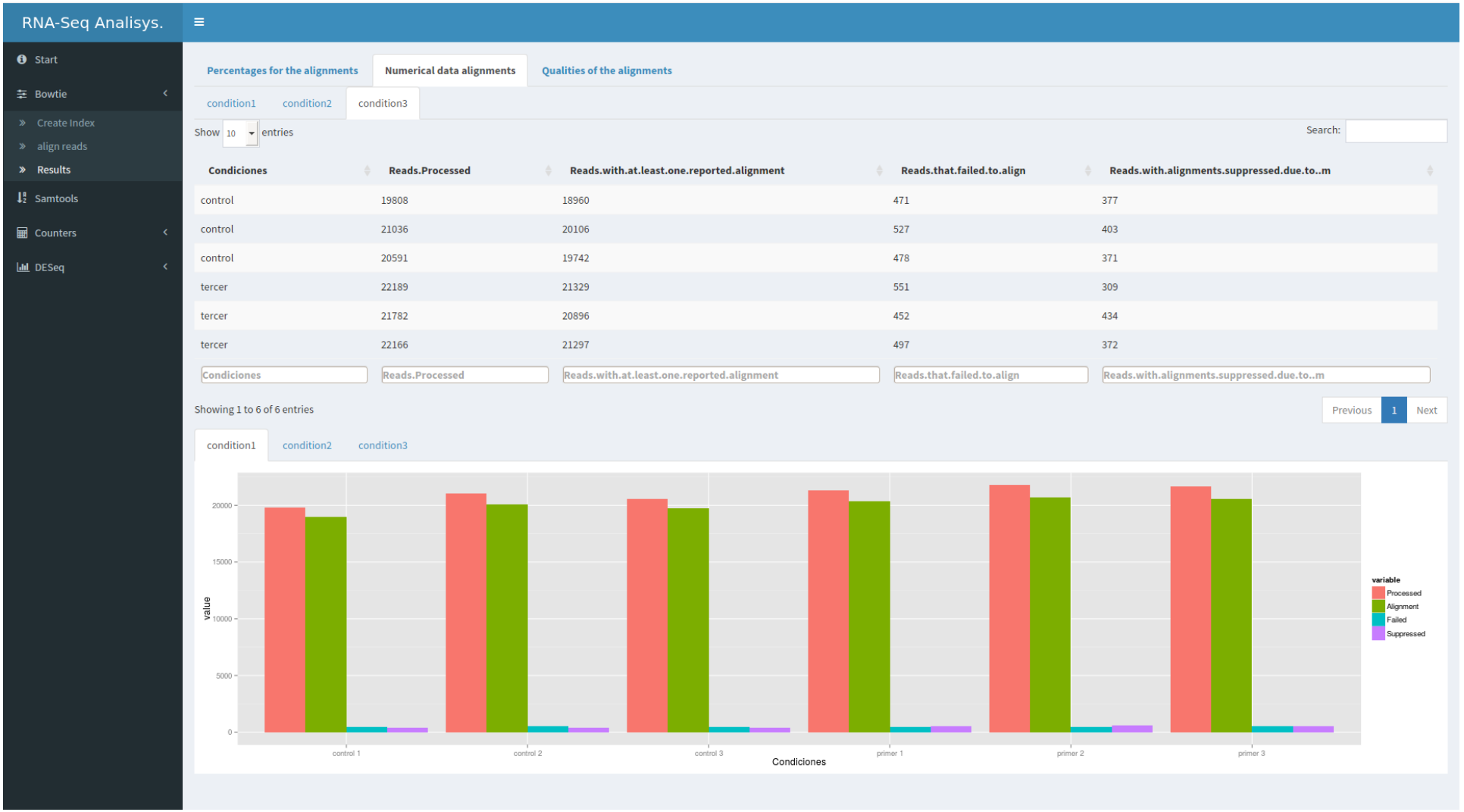


Figura B.7: Datos estadísticos de la ejecución de Bowtie en valores numéricos

Figura B.8: Datos de calidad de las muestras de las lecturas

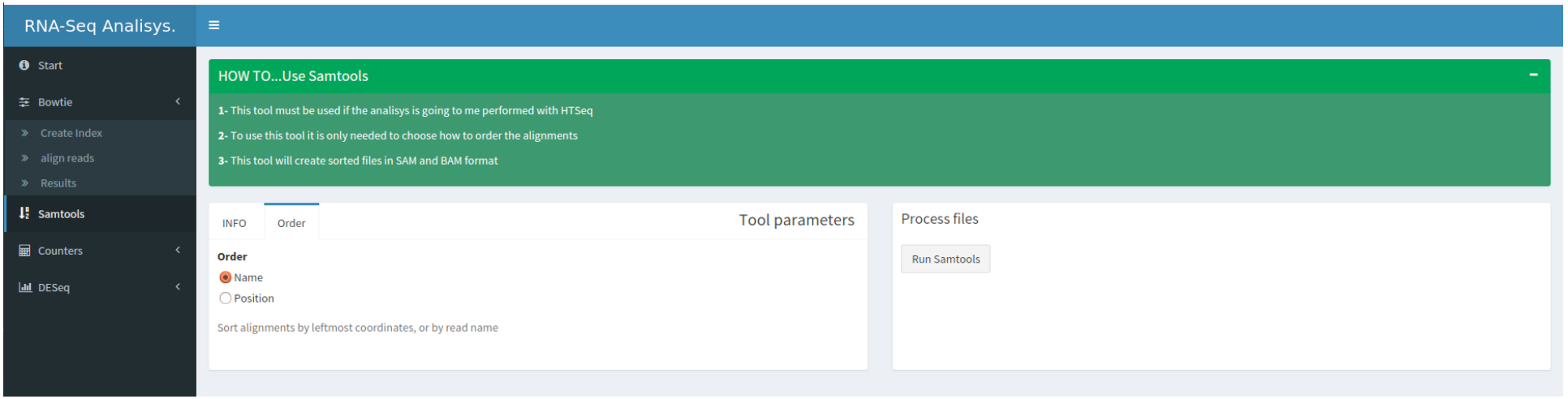


Figura B.9: Combinación de disciplinas en la bioinformática

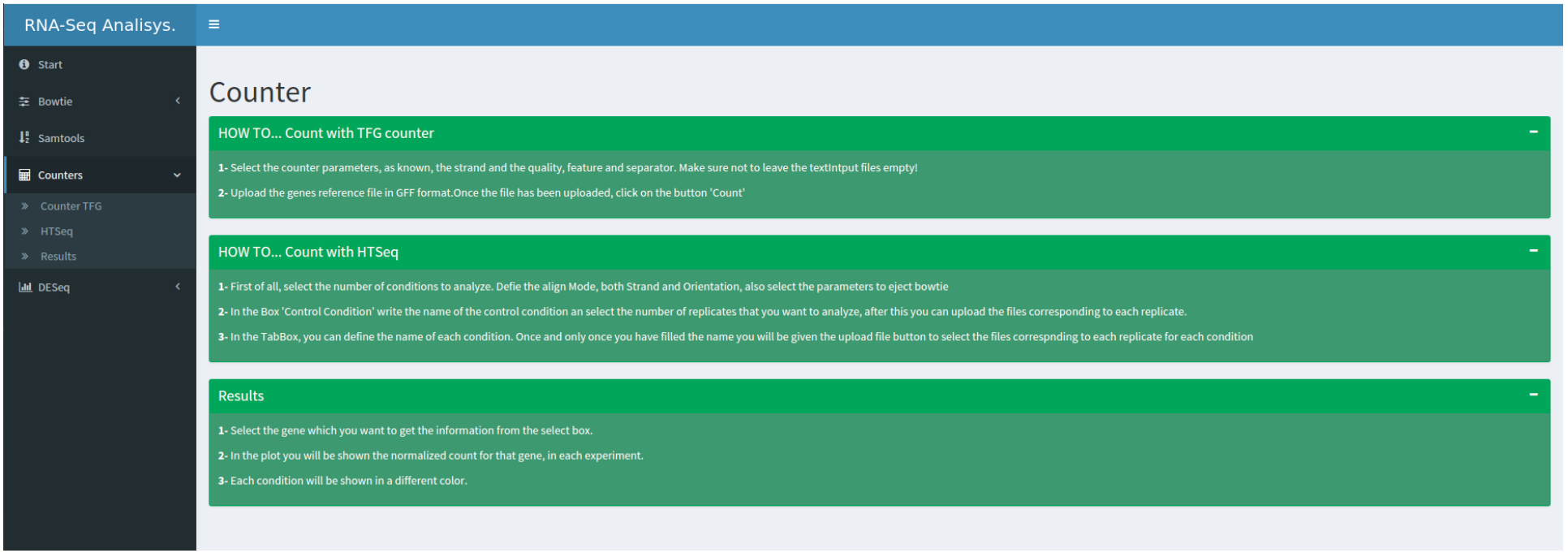


Figura B.10: Textos explicativos del uso de los contadores y los resultados generados

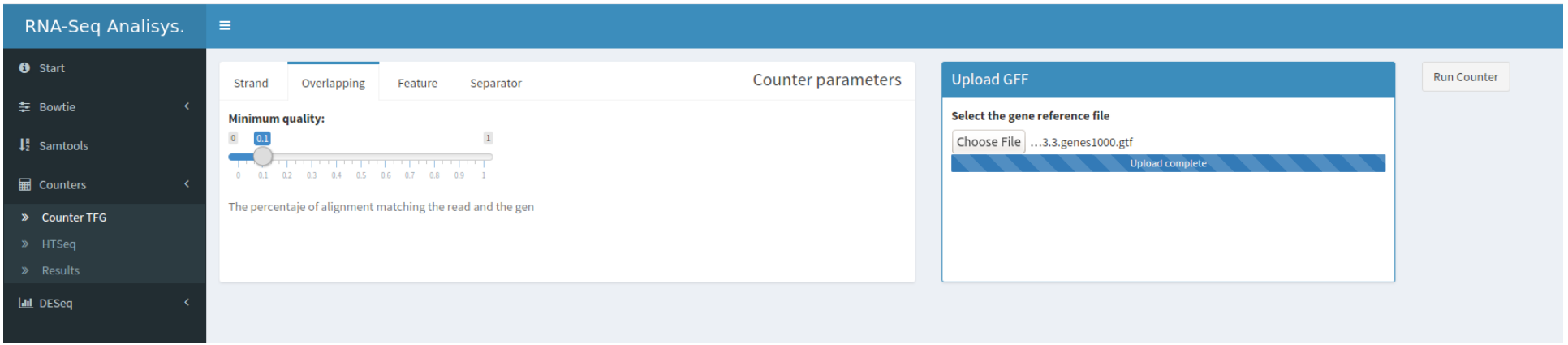


Figura B.11: Contador de genes desarrollado en el TFG

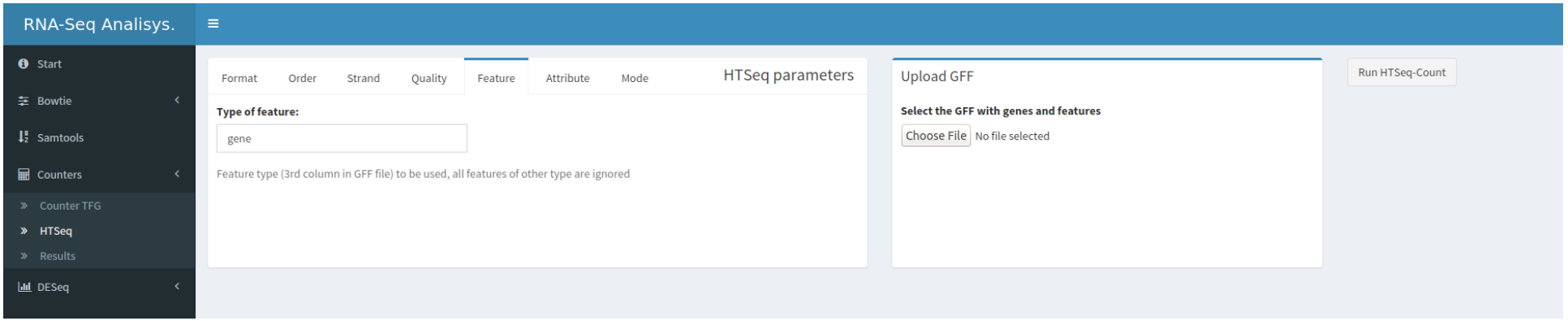


Figura B.12: Contador de genes mediante la herramienta HTSeq

Figura B.13: Ejemplo de un gen diferencialmente expresado

Figura B.14: Ejemplo de un gen no diferencialmente expresado

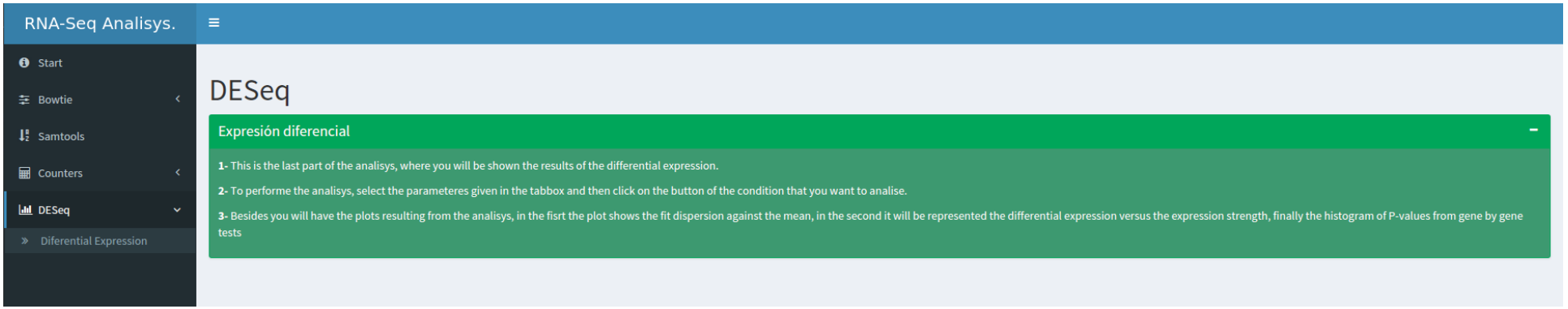


Figura B.15: Pantalla explicativa del análisis de expresión diferencial

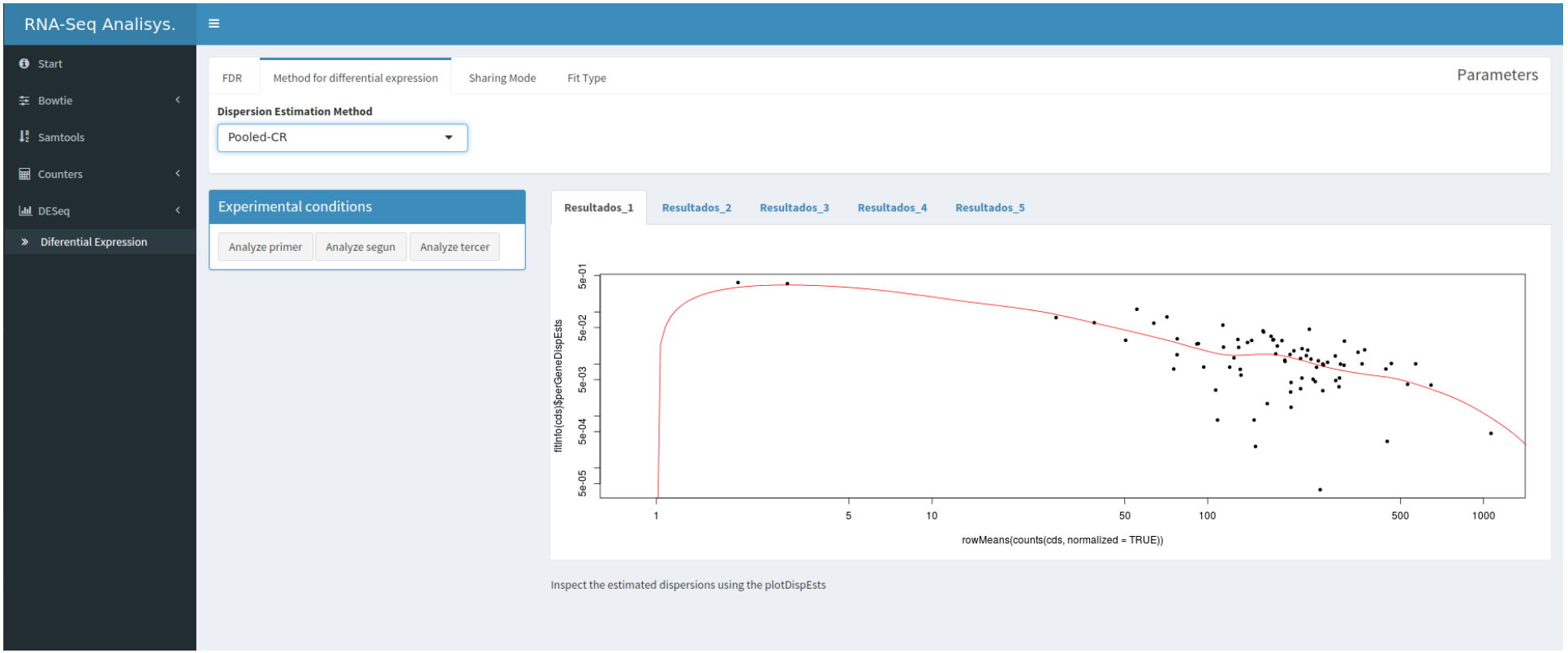


Figura B.16: Estimación de la dispersión

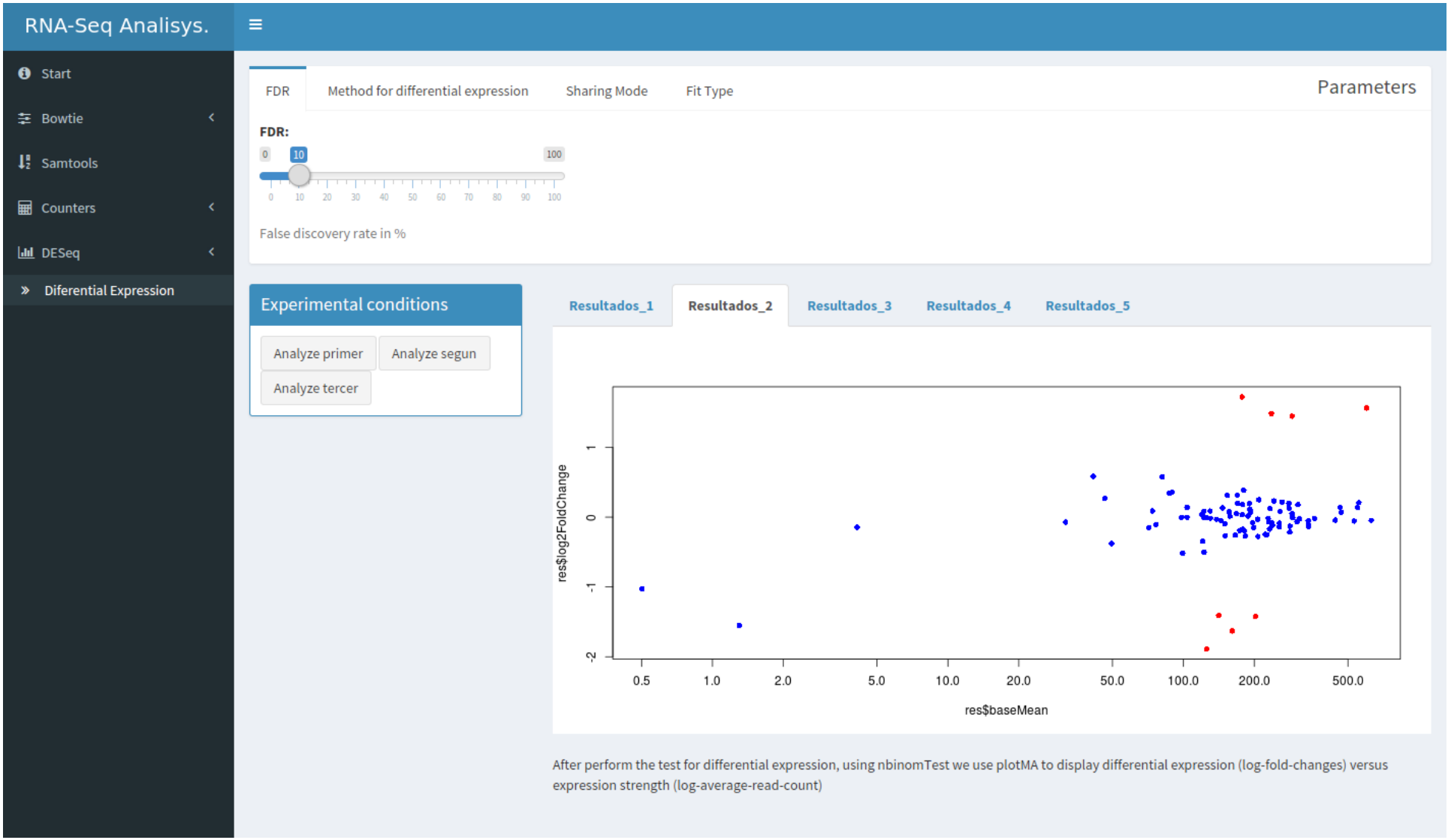


Figura B.17: Representación logarítmica del *Fold change* frente a la expresión de genes en media.

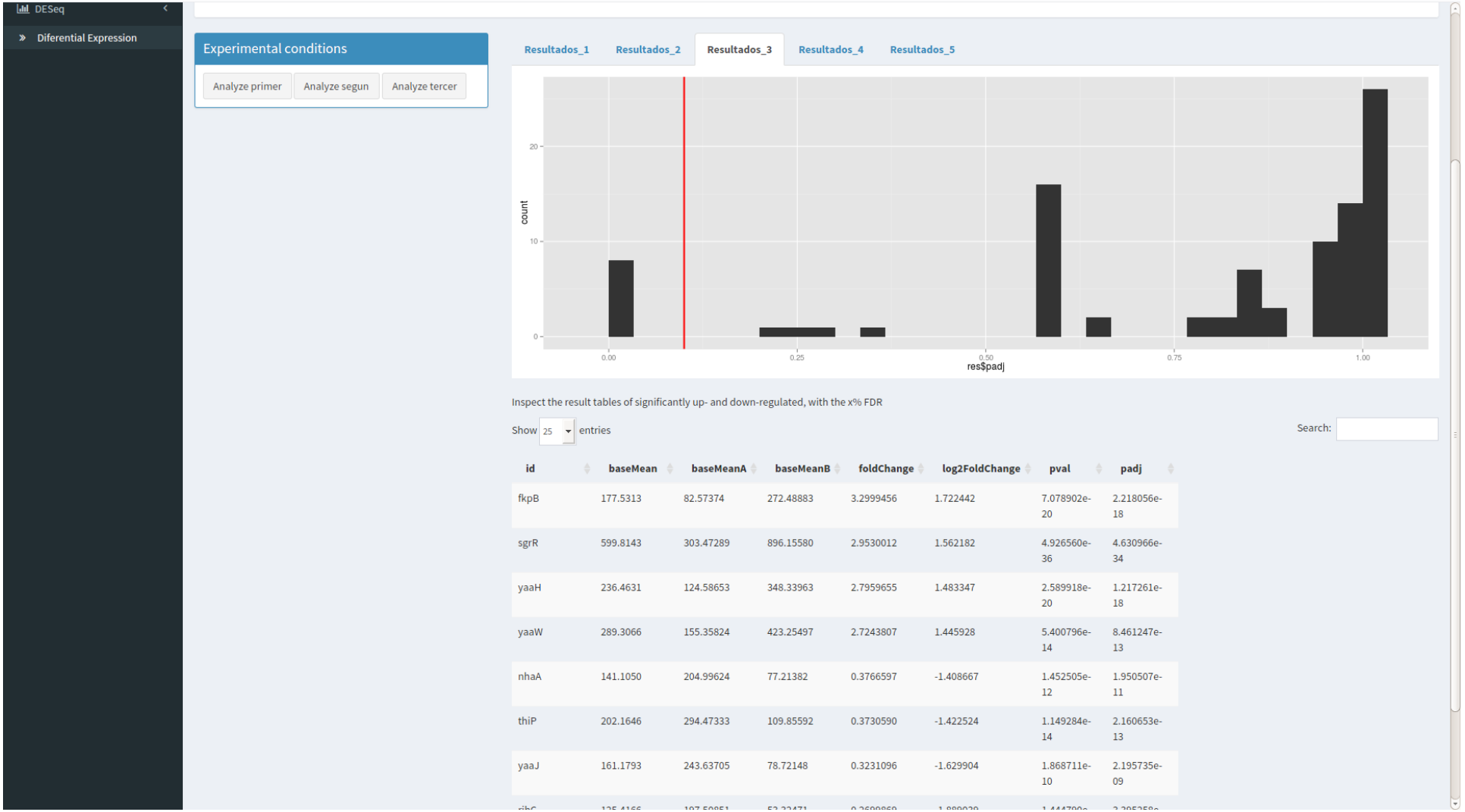


Figura B.18: Histograma de p-valores ajustados, representación de aquellos genes que están diferencialmente expresados a la izquierda de la barra vertical

C

Modelo de Datos

En este apéndice se describirán los formatos de los datos que se utilizan a lo largo del análisis de expresión diferencial.

C.1. Definición de los formatos de ficheros de datos

C.1.1. FASTA y FASTQ

En bioinformática, el formato FASTA [41] y el formato FASTQ [42] son un formato de fichero basado en texto utilizado para representar secuencias. En el caso de los ficheros FASTQ además de la representación de la secuencia se incluyen los valores de calidad, (*quality scores*), que permiten determinar cuan fiables son las lecturas realizadas sobre la secuencia original.

El fichero FASTA consta de las líneas de cabecera y datos de la secuencia. En la figura C.1 se proporciona un ejemplo de este tipo de ficheros.

Linea de cabecera: proporciona un nombre y/o un identificador único a la secuencia, y a menudo incluye información adicional. El carácter que identifica estas lineas es '>'.

Secuencias: se repersenta el genoma completo. Las secuencia pueden corresponder a secuencias de ADN, ARN, proteínas o ácidos nucleicos. En un mismo fichero FASTA podemos encontrarnos secuencias múltiples, y para cada una de ellas se han de indicar tanto la cabecera como las secuencias correspondientes. Típicamente el formato FASTA se utiliza para representar el genoma de referencia y la secuencia de nucleótidos correspondiente.

Por otra parte el fichero FASTQ se utilizan para representar las lecturas de los genomas que se quieren analizar y se compone de conjuntos de cuatro líneas:

1. Identificador de la secuencia, precedido de @. Puede contener información adicional.
2. Secuencia.
3. El carácter + y pude estar seguido del identificador de la secuencia (opcional).

```
>gi|556503834|ref|NC_000913.3| Escherichia coli str. K-12 substr. MG1655, complete genome
AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTCTGATAGCAGC
TTCTGAACTGGTTACCTGCCGTGAGTAAATTAAAATTTTATTGACTTAGGTCACTAAATACTTTAACCAA
TATAGGCATAGCGCACAGACAGATAAAAATTACAGAGTACACAACATCCATGAAACGCATTAGCACCACC
ATTACCACCACCATCACCATTACCACAGGTAACGGTGCGGGCTGACGCGTACAGGAAACACAGAAAAAAG
CCCGCACCTGACAGTGCGGGCTTTTTTTTTTCGACCAAAGGTAACGAGGTAACAACCATGCGAGTGTTGAA
GTTCGGCGGTACATCAGTGGCAAATGCAGAACGTTTTCTGCGTGTTGCCGATATTCTGGAAAGCAATGCC
AGGCAGGGGCAGGTGGCCACCGTCCTCTCTGCCCCCGCCAAAATCACCAACCACCTGGTGGCGATGATTG
AAAAAACCATTAGCGGCCAGGATGCTTTACCCAATATCAGCGATGCCGAACGTATTTTTGCCGAACTTTT
GACGGGACTCGCCGCCGCCCAGCCGGGGTTCCCGCTGGCGCAATTGAAAACTTTCGTCGATCAGGAATTT
GCCCAAATAAAACATGTCCTGCATGGCATTAGTTTGTTGGGGCAGTGCCCGGATAGCATCAACGCTGCGC
TGATTTGCCGTGGCGAGAAAATGTCGATCGCCATTATGGCCGGCGTATTAGAAGCGCGCGGTCACAACGT
TACTGTTATCGATCCGGTCGAAAAACTGCTGGCAGTGGGGCATTACCTCGAATCTACCGTCGATATTGCT
GAGTCCACCCGCCGTATTGCGGCAAGCCGCATTCCGGCTGATCACATGGTGCTGATGGCAGGTTTCACCG
CCGGTAATGAAAAAGGCGAACTGGTGGTGCTTGGACGCAACGGTTCCGACTACTCTGCTGCGGTGCTGGC
TGCCTGTTTACGCGCCGATTGTTGCGAGATTTGGACGGACGTTGACGGGGTCTATACCTGCGACCCGCGT
CAGGTGCCCGATGCGAGGTTGTTGAAGTCGATGTCCTACCAGGAAGCGATGGAGCTTTCCTACTTCGGCG
CTAAAGTTCTTCACCCCCGCACCATTACCCCCATCGCCCAGTTCCAGATCCCTTGCCTGATTAAAAATAC
CGGAAATCCTCAAGCACCAGGTACGCTCATTGGTGCCAGCCGTGATGAAGACGAATTACCGGTCAAGGGC
ATTTCCAATCTGAATAACATGGCAATGTTCAGCGTTTCTGGTCCGGGGATGAAAGGGATGGTCGGCATGG
CGGCGCGCGTCTTTGCAGCGATGTCACGCGCCCGTATTTCCGTGGTGCTGATTACGCAATCATCTTCCGA
ATACAGCATCAGTTTCTGCGTTCCACAAAGCGACTGTGTGCGAGCTGAACGGGCAATGCAGGAAGAGTTC
TACCTGGAACTGAAAGAAGGCTTACTGGAGCCGCTGGCAGTGACGGAACGGCTGGCCATTATCTCGGTGG
TAGGTGATGGTATGCGCACCTTGCGTGGGATCTCGGCGAAATTCTTTGCCGCACTGGCCCGCGCCAATAT
CAACATTGTCGCCATTGCTCAGGGATCTTCTGAACGCTCAATCTCTGTCGTGGTAAATAACGATGATGCG
ACCACTGGCGTGCGCGTTACTCATCAGATGCTGTTCAATACCGATCAGGTTATCGAAGTGTTTGTGATTG
GCGTCGGTGGCGTTGGCGGTGCGCTGCTGGAGCAACTGAAGCGTCAGCAAAGCTGGCTGAAGAATAAACA
TATCGACTTACGTGTCTGCGGTGTTGCCAACTCGAAGGCTCTGCTCACCAATGTACATGGCCTTAATCTG
GAAAACTGGCAGGAAGAACTGGCGCAAGCCAAAGAGCCGTTTAATCTCGGGCGCTTAATTCGCCTCGTGA
AAGAATATCATCTGCTGAACCCGGTCATTGTTGACTGCACTTCCAGCCAGGCAGTGGCGGATCAATATGC
CGACTTCCTGCGCGAAGGTTTCCACGTTGTCACGCCGAACAAAAAGGCCAACACCTCGTCGATGGATTAC
```

Figura C.1: FASTA: Genoma de referencia

4. Valores de calidad para la secuencia.

En la figura C.2 se muestra un ejemplo de fichero FASTQ.

```
@>read1/aaaD;mate1:43-142;mate2:210-308
GTCCGCGCCGGGCTTCGCTCACTGTTCAGGCCGGAGCCACAGACCGCCTTTGAATGGGCGGATGCTAATTACTATCTCCCGAAAGAATCCGCATACCAGG
+
?<@BDDDDH<FHDIIIIHIHIIII9ACEHH1??EFGGIGC:CCFGHHIGIG@JJHHJG#2@FH###@@@DDDDDFFDDDGE>DG@EGE@G##@@@DDDDDFFDDDGE
@>read2/aaaD;mate1:79-178;mate2:183-281
CCACAGACCGCCGTTGAATGGGCGGATGCTAATTACTATCTCCCGAAAGAATCCGCATACCAGGAAGGGCGCTGGGAAACACTGCCCTTTCAGCGGGCCA
+
@CCFFFFFH?HHHJJJJFHIFIHHGGIJJGJIJIIGHIG#0?BGHJIIJI#BCCDFFFFGHHHHIJJJJJJJIJJIIDHIGIIHIGIGIIGJGCGIIJHGHE@@@DDDDDHCFHHIIIGIIIIII
@>read3/aaaD;mate1:25-124;mate2:179-277
AACAGGCTGCGGCATTTTGTCCGCGCCGGGCTTCGCTCACTGTTCAGGCCGGAGCCACCGACCGCCGTTGAATGGGCGGATGCTAATTACTATCTCCCGA
+
@@@DFFFDHDHHHIBBEFIGGIHIIIIGGHHIIIGICDGCGHIGHIIIIIICCBFFFFFHHGHHJJJJHHAFHIJHJIIJJJJJFIIJJJJJIJJIJJJJJJJIJIIFHIIJJJJJJJJJGII#0:DHG#
@>read4/aaaD;mate1:22-121;mate2:206-304
GTTAACAGGCTGCGGCANTTTGTCCGCGCCGGGCTTCGCTCACTGTTCAGGCCGGAGCCACAGACCGCCGTTGAATGGGCGGATGCTAATTACTATCTCC
+
BCCFFDDFHHHHHJJJJJJJJHIJIJJJJJJIJJJJIIJ#0?FHHIJJJJCCCFFFFFHGHGHIJIJJJJIIJHIJJJJJJ####################CCCFFFFFHHHHHJJJJJFFHFHHF
@>read5/aaaD;mate1:14-113;mate2:144-242
ACAGTCAGGTTAACAGGATGCGGCATTTTGTCCGCGCCGGGCTTCGCTCACTGTTCAGGCCGGAGCCACAGACCGCCGTTGAATGGGCGGATGCTAATTA
+
@CCFFFFDHHHHHJJJJFHIIJJIIHHFFFFFFEEDEEEDDEDDFDFFFHHHHHJIJHGIJEDDDDD@@@@FDFFFHHHHHJIJHGIJHIJJ######################
@>read6/aaaD;mate1:33-132;mate2:194-292
GCGGCATTTTGTCCGCGCCGGGCTTCGCTCACTGTTCAGGCCGGAGCCACAGACCGCCGTTGAATGGGCGGATGCTAATTACTATCTCCCGAAAGAATCC
+
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

Figura C.2: FASTQ: Lecturas(*reads*) de un experimento

C.1.2. SAM *Sequence Alignment/Map*

Este tipo de ficheros recoge la información referente a las secuencias y los pares alineación (SAM) [1]. Se componen de una cabecera y un conjunto de alineaciones. En la figura C.3 se puede ver un ejemplo de estos ficheros.

La parte correspondiente a la cabecera se compone de pares *ETIQUETA:VALOR*, cada una de ellas precedidas de @. Los valores obligatorios se describen el a la tabla C.1.

```
@HD   VN:1.0 SO:coordinate
@SQ   SN:seq1         LN:5000
@SQ   SN:seq2         LN:5000
@CO   Example of SAM/BAM file format.
B7_591:4:96:693:509   73      seq1    1       99      36M     *       0       0       CACTAGTGGCTCATTGTAAATGTGTC
CTCG    <<<<<<<<<<<<<<<;<<<<<<<<<5<<<<<;:<;7     MF:i:18 Aq:i:73 NM:i:0  UQ:i:0  H0:i:1  H1:i:0
EAS54_65:7:152:368:113          73      seq1    3       99      35M     *       0       0       CTAGTGGCTCATTGTAAAT
GTGTGGTTTAACTCGT        <<<<<<<<<<0<<<<655<<7<<<:9<<3/:<6):    MF:i:18 Aq:i:66 NM:i:0  UQ:i:0  H0:i:1  H1:i:0
EAS51_64:8:5:734:57   137     seq1    5       99      35M     *       0       0       AGTGGCTCATTGTAAATGTGTGGTTT
CC      <<<<<<<<<<<7;71<<;<;;<7;<<3;);3*8/5    MF:i:18 Aq:i:66 NM:i:0  UQ:i:0  H0:i:1  H1:i:0
B7_591:1:289:587:906  137     seq1    6       63      36M     *       0       0       GTGGCTCATTGTAATTTTTTGTTTTA
CTCTTCTCT    (-&----,----)-)-),'--)---',+-,),'"*,       MF:i:130        Aq:i:63 NM:i:5  UQ:i:38 H0:i:0  H1:i:0
EAS56_59:8:38:671:758137        seq1    9       99      35M     *       0       0       GCTCATTGTAAATGTGTGGTTTAACT
CGTCCATGG   <<<<<<<<<<<<<<<;<;7<<<<<<<<7<<;:<5%      MF:i:18 Aq:i:72 NM:i:0  UQ:i:0  H0:i:1  H1:i:0
EAS56_61:6:18:467:28173         seq1    13      99      35M     *       0       0       ATTGTAAATGTGTGGTTTAACTCGTC
GCCCA       <<<<<<<<;<<<8<<<<<;8:;6/686&;(16666 MF:i:18 Aq:i:39 NM:i:1  UQ:i:5  H0:i:0  H1:i:1
EAS114_28:5:296:340:699         137     seq1    13      99      36M     *       0       0       ATTGTAAATGTGTGGTTTAAC
TCGTCCATGGCCCAG <<<<<;<<<;<<;<;<<<<<<<<<<8<8<3<8;<;<0;         MF:i:18 Aq:i:73 NM:i:0  UQ:i:0  H0:i:1  H1:i:0
B7_597:6:194:894:408  73      seq1    15      99      35M     *       0       0       TGTAAATGTGTGGTTTAACTCGTCCA
C       <<<<<<<<<7<<;<<<<;<<<7;;<<<*,;;572<  MF:i:18 Aq:i:43 NM:i:1  UQ:i:9  H0:i:0  H1:i:1
EAS188_4:8:12:628:97389         seq1    18      75      35M     *       0       0       AAATGTGTGGTTTAACTCGTCCATGC
CCAGCATT    ==;=;;;;;;====;=;===;=======;==;===     MF:i:64 Aq:i:0  NM:i:0  UQ:i:0  H0:i:1  H1:i:0
EAS51_66:7:68:402:50  137     seq1    22      99      35M     *       0       0       GTGTGGTTTAACTCGTCCATGGCCC
AGCATTTGGG <<<<<<<<<<<<<<:<<<9<6;9;;&697;7&<55         MF:i:18 Aq:i:66 NM:i:1  UQ:i:5  H0:i:1  H1:i:0
```

Figura C.3: SAM: Alineaciones con el genoma de referencia

Tag	Valor	Descripción
@HD	VN	Format version.
@SQ	SN	Defines the alignment sorting order.
	LN	Reference sequence name and length.
@RG	ID	Read group identifier
@PG	ID	Program record identifier.

Tabla C.1: Identificadores obligatorios de la cabecera de los ficheros de alineaciones

Por otra parte el cuerpo de estos ficheros contienes información sobre las alineaciones y elementos que no han conseguido ser alineados.Se sigue una estructura en campos separados por tabuladores. Los campos obligatorios son los siguientes:

Bit		Description
1	0x1	template having multiple segments in sequencing
2	0x2	each segment properly aligned according to the aligner
4	0x4	segment unmapped
8	0x8	next segment in the template unmapped
16	0x10	SEQ being reverse complemented
32	0x20	SEQ of the next segment in the template being reverse complemented
64	0x40	the first segment in the template
128	0x80	the last segment in the template
256	0x100	secondary alignment
512	0x200	not passing filters, such as platform/vendor quality controls
1024	0x400	PCR or optical duplicate
2048	0x800	supplementary alignment

Figura C.4: Bitwise flag description [1]

1. QNAME: Identificador de la secuencia. Todas aquellas que tengan este campo idéntico se analizarán como que proceden de la misma lectura. Sirve para determinar si una lectura alinea más de una vez con el genoma de referencia y para identificar las lecturas *paired-end*.

2. FLAG: Número que queda determinado por una combinación de bits. Cada posición proporciona la información mostrada en la figura C.4

3. RNAME: Nombre o identificador del genoma de referencia asociado a la secuencia de la alineación.
4. POS: Primera coincidencia en la dirección de alineación del *read* con el genoma de referencia.
5. MAPQ: Calidad de la alineación con respecto al genenoma de referencia.
6. CIGAR: Secuencia de caracteres que determina el numero de *matches*, sustituciones, borrados o errores de la lectura con respecto al genoma de referencia en el caso de las lecturas *paired-end*.
7. RNEXT: RNAME del par asociado a la alineación en el caso de las lecturas *paired-end*.
8. PNEXT: Posición del par asociado a la alineación.
9. TLEN: Número de bases con los que a alineado la lectura.
10. SEQ: Segmento de la secuencia que ha alineado con el genoma de referencia.
11. QUAL: Campo de calidad asociado a la secuencia.

En caso de haber algun campo vacío o no especificado se ha de completar dicho campo con el símbolo "*".

C.1.3. Formato GFF

Es un formato de ficheros que se utiliza para la descripción de genes y otras características de las secuencias tanto del ADN como del ARN o proteínas. Se denominan GFF o GTF. Al igual que el formato SAM se compone de campos separados por tabuladores y tiene la particularidad de que los campos vacíos se determinan con el símbolo ".". En la figura C.5 se proporciona un ejemplo del formato.

```
NC_000913.3  RefSeq  Coding gene  190    255    .  +  .  name=thrL;product="thr operon leader peptide"
NC_000913.3  RefSeq  Coding gene  337    2799   .  +  .  name=thrA;product="fused aspartokinase I and homoserine dehydrogenase I"
NC_000913.3  RefSeq  Coding gene  2801   3733   .  +  .  name=thrB;product="homoserine kinase"
NC_000913.3  RefSeq  Coding gene  3734   5020   .  +  .  name=thrC;product="threonine synthase"
NC_000913.3  RefSeq  Coding gene  5234   5530   .  +  .  name=yaaX;product="predicted protein"
NC_000913.3  RefSeq  Coding gene  5683   6459   .  -  .  name=yaaA;product="peroxide resistance protein, lowers intracellular iron"
NC_000913.3  RefSeq  Coding gene  6529   7959   .  -  .  name=yaaJ;product="predicted transporter"
NC_000913.3  RefSeq  Coding gene  8238   9191   .  +  .  name=talB;product="transaldolase B"
NC_000913.3  RefSeq  Coding gene  9306   9893   .  +  .  name=mog;product="molybdochelatase incorporating molybdenum into molybdopterin"
NC_000913.3  RefSeq  Coding gene  9928   10494  .  -  .  name=yaaH;product="inner membrane protein, Grp1_Fun34_YaaH family"
NC_000913.3  RefSeq  Coding gene  10643  11356  .  -  .  name=yaaW;product="conserved protein, UPF0174 family"
NC_000913.3  RefSeq  Coding gene  11382  11786  .  -  .  name=yaaI;product="conserved protein, UPF0412 family"
NC_000913.3  RefSeq  Coding gene  12163  14079  .  +  .  name=dnaK;product="chaperone Hsp70, co-chaperone with DnaJ"
NC_000913.3  RefSeq  Coding gene  14168  15298  .  +  .  name=dnaJ;product="chaperone Hsp40, co-chaperone with DnaK"
```

Figura C.5: GFF: Referencia a los genes de Escherichia Coli

Los campos de los que consta son:

1. Seqname: nombre del cromosoma al que hace referencia.
2. Source: nombre del programa que ha generado esta característica.
3. Feature: tipo de la característica (gen, exon, variación).
4. Start: posición de inicio de la característica o gen.
5. End: posición final.

6. Strand: define si la alineación se produce en + (*forward*) o - (*reverse*).

7. Attribute: lista de pares etiqueta-valor, separados por un "*;*"

C.1.4. Matriz de conteo de genes

Es el formato de salida de la herramienta de conteo utilizada que genera una matriz de conteo de genes. En la figura C.6 se muestra un ejemplo de matriz con 6 experimentos.

La matriz se compone de filas, las cuales se corresponden con cada una de las características del GFF que se hayan tenido en cuenta a la hora de realizar el conteo. Cada columna es un experimento.

El número corresponde a la cantidad de coincidencias posicionales que se han encontrado en el SAM analizado. En la sección 5.2.1 se puede encontrar una descripción detallada de dichas coincidencias.

aaaD	3	1	4	0	19	0
aaaE	13	12	28	4	17	16
aaeA	228	166	215	153	211	243
aaeB	298	238	272	248	281	173
aaeR	684	547	588	407	513	718
aaeX	119	169	165	161	117	136
aas	1408	1147	1314	1448	1285	1478
aat	828	834	659	803	802	923
abgA	42	24	5	24	10	20
abgB	34	30	20	58	44	22
abgR	110	153	70	184	91	157
abgT	58	27	43	5	50	13
abrB	182	203	149	231	242	221

Figura C.6: Resultado del contador: una matriz de conteo

D

Herramientas: Alineadores, Samtools y DESeq

En este apéndice se incluye el *timeline* de alineadores por fecha de creación, una breve explicación de la herramienta Samtools, necesaria para el análisis de expresión diferencial cuyo *pipeline* incluye la herramienta HTSeq. Por otra parte se incluye el código implementado para el análisis de los datos a partir de las funciones del paquete DESeq.

D.1. Timeline de alineadores

La figura D.1 muestra múltiples alineadores que se han creado en los últimos años. Se puede apreciar un aumento considerable del numero de alineadores a partir del año 2007.

D.2. Orden e indexación de alineaciones: Samtools

Es una herramienta de código abierto utilizada también en el análisis de datos RNA-seq. Se utiliza para procesar las alineaciones, conversiones entre dos tipos de formatos de alineaciones como puede ser SAM y BAM, y ordenar o indexar las alineaciones. En este trabajo se utiliza como paso previo al análisis necesario para el uso de la herramienta HTSeq, ya que los datos generados por Bowtie, han de ser ordenados.

D.3. DESeq: Análisis de expresión diferencial y representación de resultados

Código implementado para el análisis de expresión diferencial a partir de funciones de la librería DESeq de *Bioconductor* de R.

```
#lectura de los datos de entrada
countTable <- read.table("counted-genes.txt", sep="\t",
```

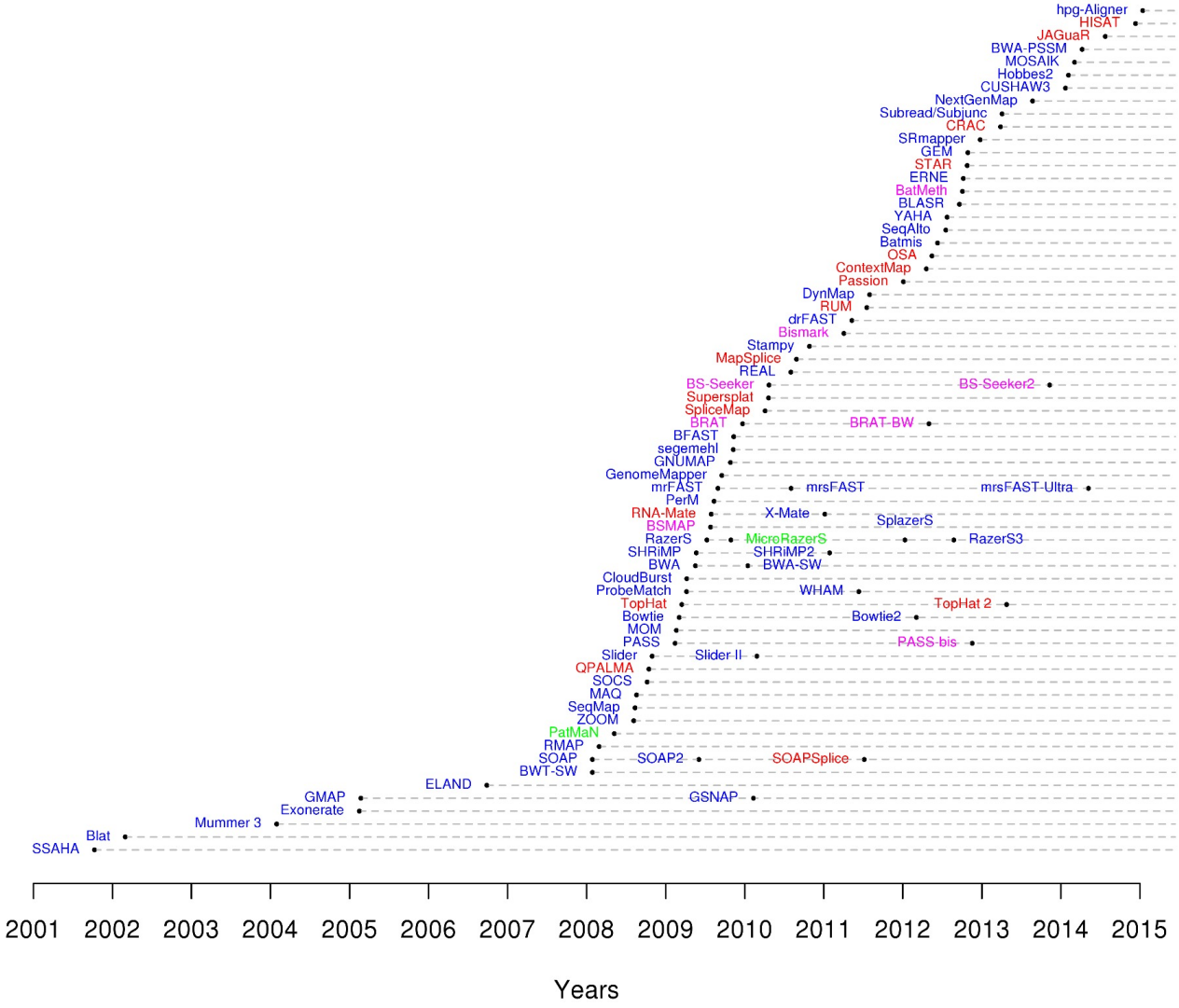


Figura D.1: Evolución temporal de alineadores NGS[2].

```
                        header = FALSE, row.names = 1)

#estructurar labla en formato dataframe
datosFrame <-data.frame(row.names = replicas,
                        c_list_conds, c_list_tipos)
#Seleccionar los datos en formato paired o
#single y estructurarlos en un nuevo dataframe

#paired-end
pairedSamples <- datosFrame$c_list_tipos == 1
#single-end
singleSamples <- datosFrame$c_list_tipos == 2
countPairedTable <-countTable[,pairedSamples]
conds <- datosFrame$c_list_conds[pairedSamples]
```

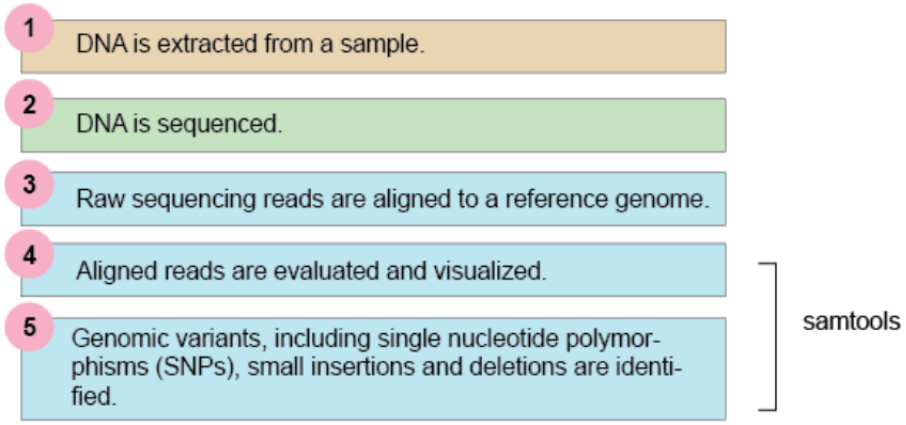


Figura D.2: Utilización de la herramienta Samtools en el análisis [3]

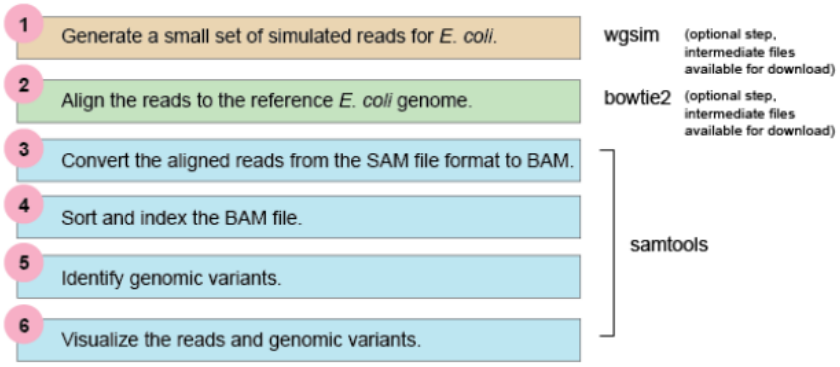


Figura D.3: Distintos usos de la herramienta Samtools [3]

```
#construccion de un dataSet para el analisis RNA-seq con DESeq
cds<- newCountDataSet(countPairedTable,conds)

#estimacion del size factor de los datos para su posterior normalizacion
cds<- estimateSizeFactors(cds)
table_normalized =counts(cds, normalized= TRUE    )
#Con los parametros: fitType, sharingMode, method.
#Estos los determinara el ususario
cds <- estimateDispersions( cds, fitType="parametric",
                            sharingMode= "fit-only", method = "pooled")

#REPRESENTACIoN DE RESULTADOS
# Estimaciones de dispersion por cada gen,
#junto con la relacion ajustada de la media-dispersion
plotDispEsts <- function( cds ){
  plot(
    rowMeans(counts(cds,normalized=TRUE)),
```

```
    fitInfo(cds)$perGeneDispEsts,
    pch = ".", log="xy")
  xg<-10^seq(-.5, 5, length.out = 300)
  lines(xg, fitInfo(cds)$dispFun(xg),col="red")
}
output$primero <-renderPlot({
  plotDispEsts(cds)
})

# Test con la binomial negativa, sobre los datos,
#la condicion de control y la condicion experimental
res <- nbinomTest( cds, paste0(input$control_condition),
                   paste0(input$exp_condition ))

# Repesentacion de la media de los conteos de todos
#los experimentos normallizados frente al log del 'fold-change'
# Representacion de los valores en ROJO para
#aquellos genes diferencialmente expresados
plotDE <- function( res, valor )
  plot(
    res$baseMean,
    res$log2FoldChange,
    log="x", pch=20, cex=.3,
    col = ifelse( res$padj < valor, "red", "blue" ) )
output$segundo <-renderPlot({
  val_cota = ((as.numeric(input$cota))/100)
  plotDE(res,val_cota)
})

#Histograma de los p-valores ajustados.
#Se incluye linea vertical separando por el FDR
output$tercero <-renderPlot({
  val_cota = (as.numeric(input$cota))/100
  env = environment()
  ggplot(res, aes(x=res$padj), environment= env) +
    geom_histogram() +
    geom_vline(aes(xintercept=val_cota),
               color = "red", size=1, show_guide=T)
})

#Muestra en tabla de los genes diferencialmente expresados
output$dif_expresed <- renderDataTable({
  resSig = res[which(res$padj < 0.1),]
  print(res$ig)
  resSig[ order(resSig$log2FoldChange, decreasing=TRUE), ]
})

#Representacion de los p-valores frente al log del 'fold-change'
output$cuarto <- renderPlot({
```

```
  env = environment()
  a<-ggplot(res, aes(x=pval, y=log2FoldChange),
            environment= env) + geom_point()+

    xlab("P-value") +

    ylab("Log_2 Fold Change") +

    theme(axis.text.x = element_text(angle = 45, hjust = 1))
  plot (a)
})

#Output de la tabla de resultados obtenidos mediante el analisis.
output$quinto <- renderDataTable({
  df_res<-data.frame(res)
  print(df_res)
  names(df_res) = c("id", "baseMean", "baseMeanA",
                    "baseMeanB", "foldChange",
                    "log2FoldChange" ,"pval", "padj")
  df_res
})
```

E

Organización y secuencia temporal

En este anexo se han incluido el diagrama de Gantt de las tareas realizadas a lo largo del del desarrollo del TFG (ver figura E.1). Además se incluye una tabla en al que se indican la fecha de inicio y fin de la tarea así como el número de horas dedicadas a cada una de ellas (ver figura E.2).

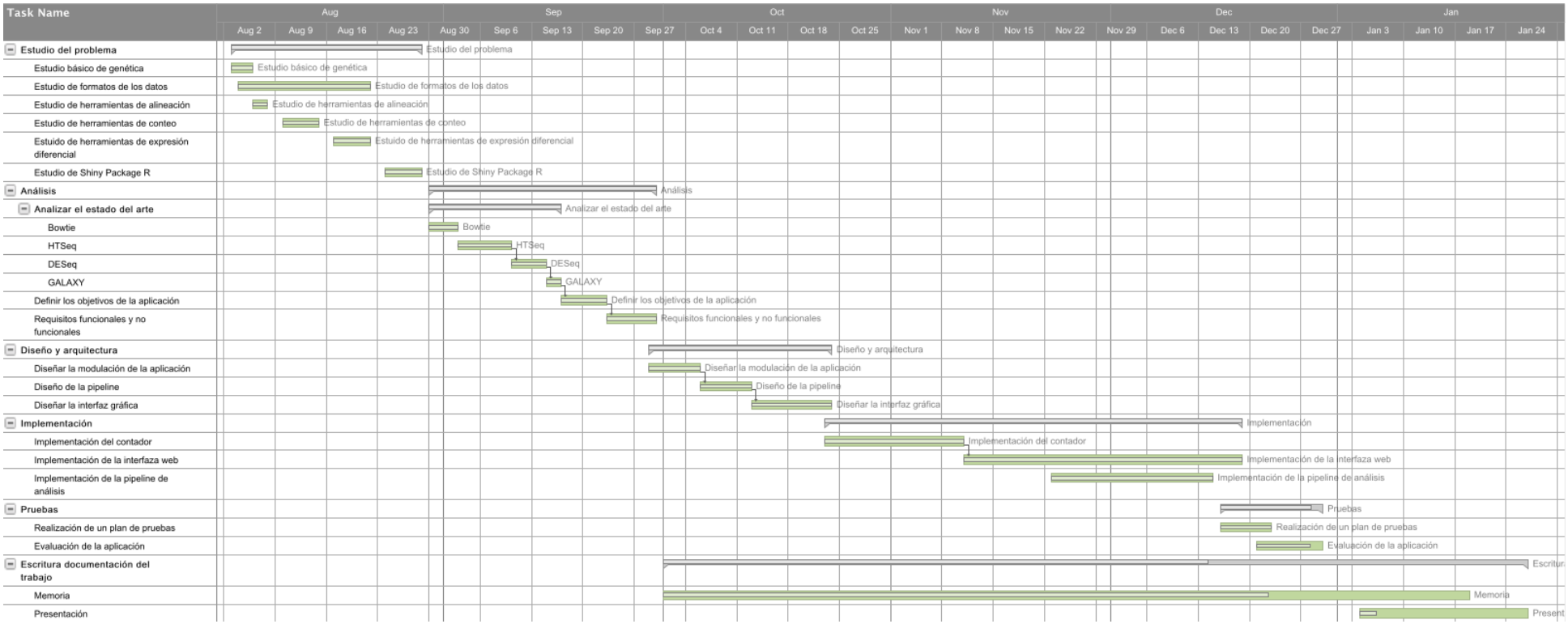


Figura E.1: Diagrama de Gantt del desarrollo del TFG

Task Name	Start Date	End Date	Duration	% Complete	Assigned To	Hours dedicated
Estudio del problema	**08/03/15**	**08/28/15**	**20d**	**100%**	Susana	**30 aprox**
Estudio básico de genética	08/03/15	08/05/15	3d	100%	Susana	
Estudio de formatos de los datos	08/04/15	08/21/15	14d	100%	Susana	
Estudio de herramientas de alineación	08/06/15	08/07/15	2d	100%	Susana	
Estudio de herramientas de conteo	08/10/15	08/14/15	5d	100%	Susana	
Estuido de herramientas de expresión diferencial	08/17/15	08/21/15	5d	100%	Susana Hdez	
Estudio de Shiny Package R	08/24/15	08/28/15	5d	100%	Susana	
Análisis	**08/30/15**	**09/29/15**	**23d**	**100%**	Susana	**50 aprox**
Analizar el estado del arte	08/30/15	09/16/15	14d	100%	Susana	
Bowtie	08/30/15	09/02/15	4d	100%	Susana	
HTSeq	09/03/15	09/10/15	5.25d	100%	Susana	
DESeq	09/10/15	09/14/15	2.75d	100%	Susana	
GALAXY	09/15/15	09/16/15	2d	100%	Susana	
Definir los objetivos de la aplicación	09/17/15	09/23/15	4.25d	100%	Susana	
Requisitos funcionales y no funcionales	09/23/15	09/29/15	4.75d	100%	Susana	
Diseño y arquitectura	**09/29/15**	**10/23/15**	**19d**	**100%**	Susana	**70 aprox**
Diseñar la modulación de la aplicación	09/29/15	10/05/15	5d	100%	Susana	
Diseño de la pipeline	10/06/15	10/12/15	5d	100%	Susana	
Diseñar la interfaz gráfica	10/13/15	10/23/15	9d	100%	Susana	
Implementación	**10/26/15**	**12/23/15**	**43d**	**100%**	Susana	**100 aprox**
Implementación del contador	10/26/15	11/10/15	12d	100%	Susana	
Implementación de la interfaza web	11/11/15	12/23/15	31d	100%	Susana	
Implementación de la pipeline de análisis	11/23/15	12/14/15	16d	100%	Susana	
Pruebas	12/16/15	12/30/15	11d	87%	Susana	**20 aprox**
Realización de un plan de pruebas	12/16/15	12/21/15	4d	900%	Susana	
Evaluación de la aplicación	12/22/15	12/30/15	7d	80%	Susana	
Escritura documentación del trabajo	10/01/15	01/29/16	87d	62%	Susana	**30 aprox**
Memoria	10/01/15	01/18/16	78d	75%	Susana	
Presentación	01/04/16	01/29/16	20d	10%	Susana	

Figura E.2: Tabla de horas dedicadas da las tareas.

F

Workflow y Modo de conteo

En este anexo se muestran los *pipeline* del análisis de expresión diferencial, tanto del análisis que se ha decidido implementar en el TFG (ver figura F.1) como el *workflow* del estado del arte proporcionado por el artículo [4] (ver figura F.2). También se incluyen las formas de conteo que utilizan tanto el contador implementado (ver figura F.3) como el que utiliza HTSeq en su herramienta *htseq-count* (ver figura F.4).

F.1. *Workflow* para la aplicación y del estado del arte

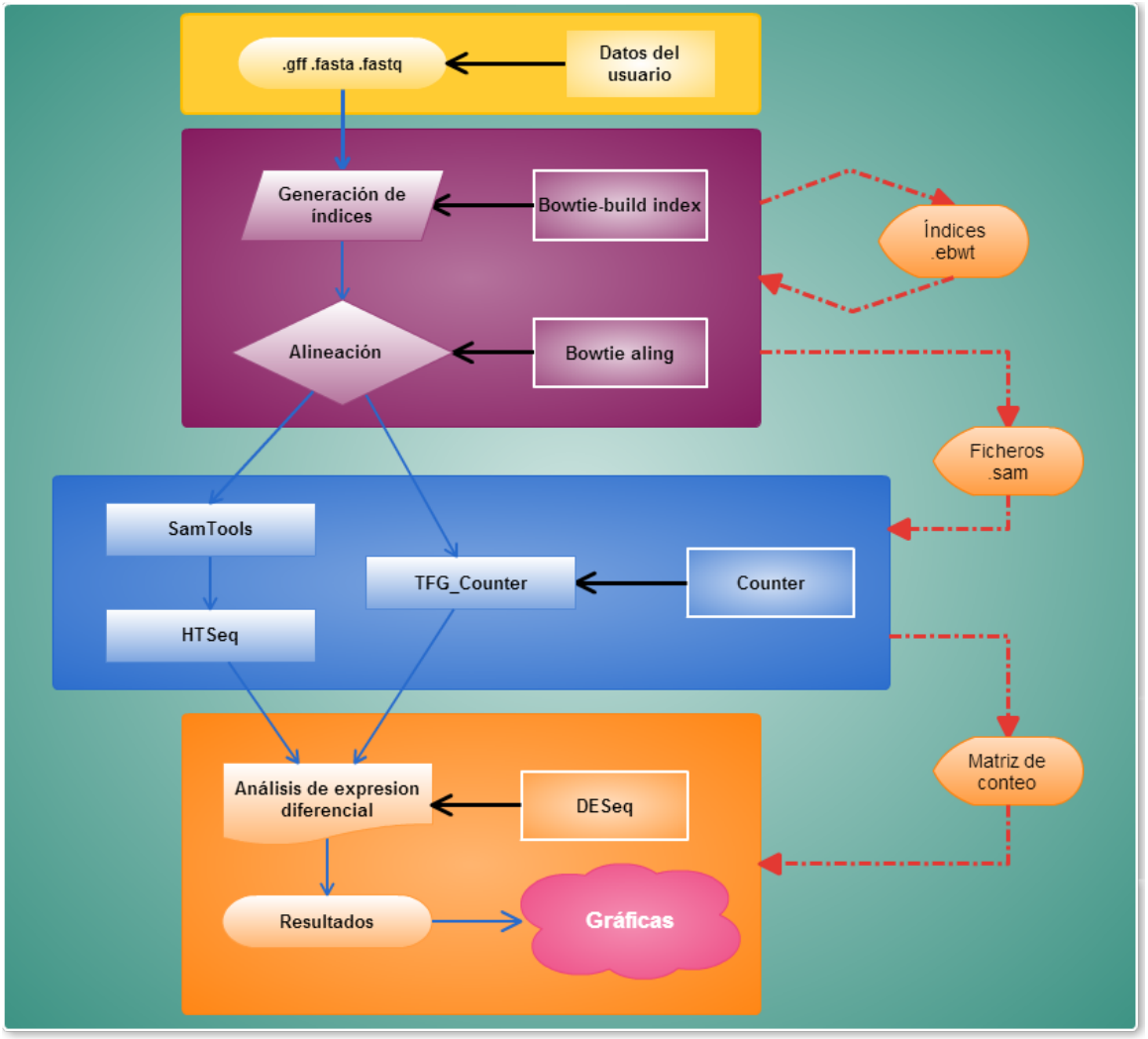


Figura F.1: Workflow de la aplicación

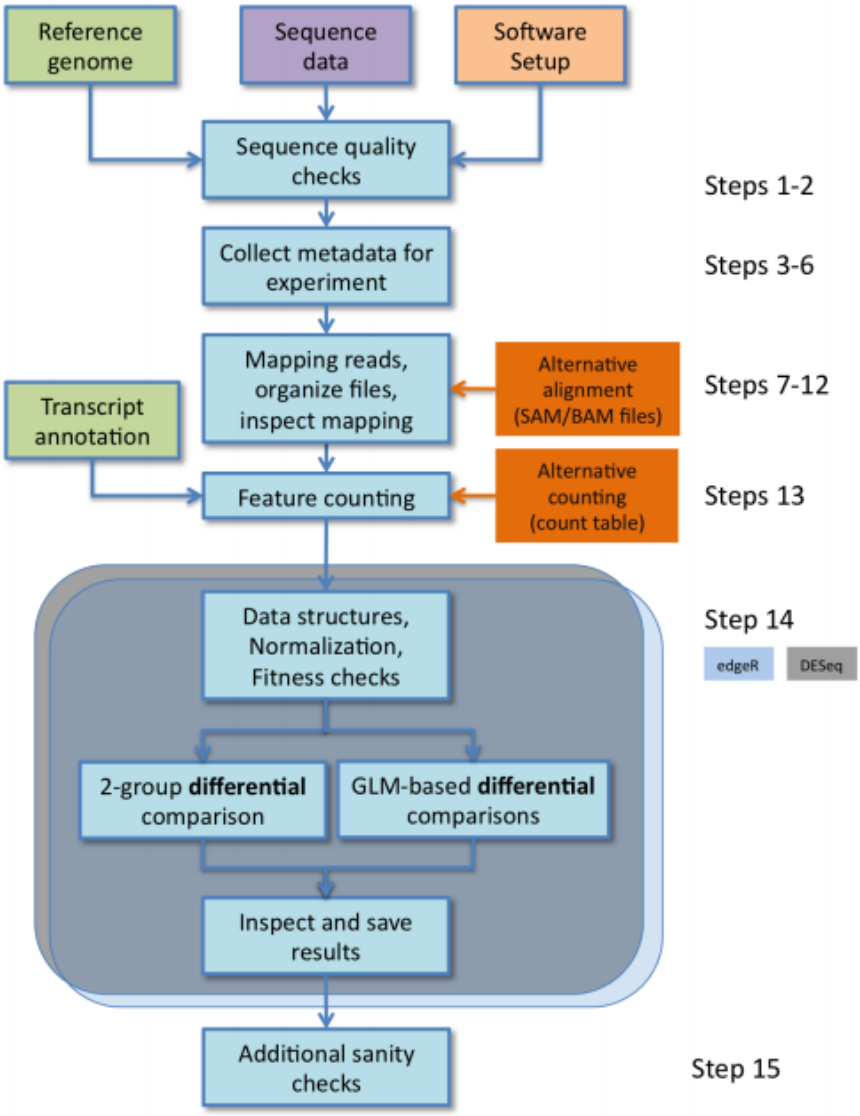


Figura F.2: *Workflow* o *pipeline* del análisis de expresión diferencial [4]. Estado del arte.

F.2. Modo de conteo de genes del contador implementado y del HTSeq

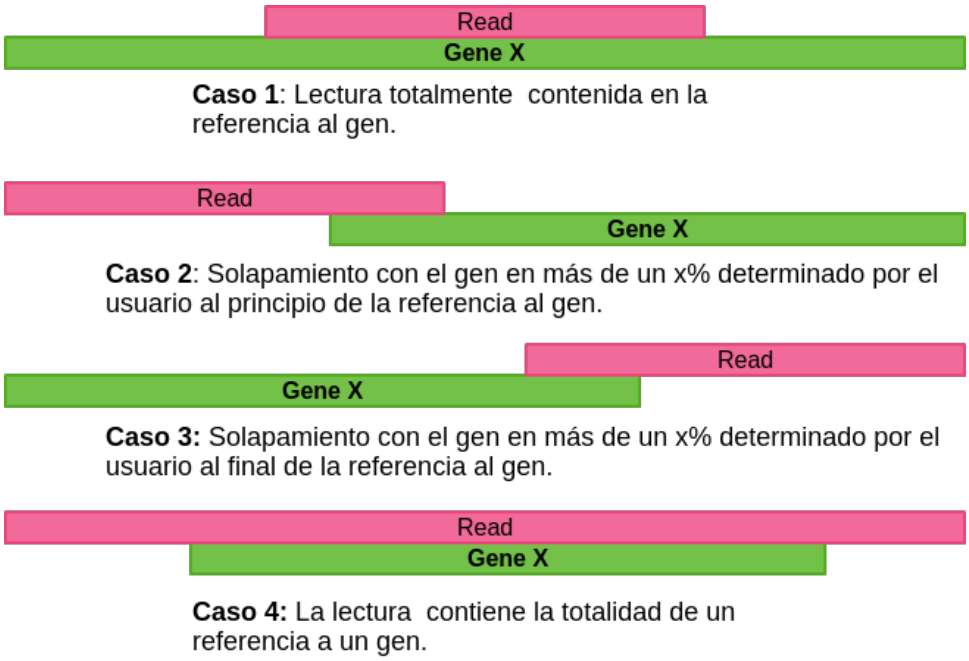


Figura F.3: Casos de conteo del contador implementado para cada gen.

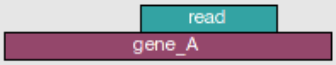
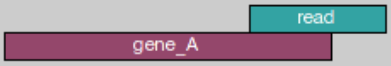
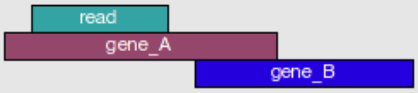
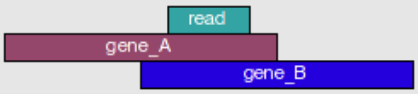
	union	intersection _strict	intersection _nonempty
	gene_A	gene_A	gene_A
	gene_A	no_feature	gene_A
	gene_A	no_feature	gene_A
	gene_A	gene_A	gene_A
	gene_A	gene_A	gene_A
	ambiguous	gene_A	gene_A
	ambiguous	ambiguous	ambiguous

Figura F.4: Condiciones de conteo del HTSeq [5].

G

Maquetas y Resultados gráficos

En este anexo se muestran las representaciones de las maquetas descritas en el capítulo 5. Además, se incluyen las imágenes de las gráficas con los resultados del análisis de expresión diferencial.

G.1. Maquetas

A continuación se proporcionan las visualizaciones de las maquetas descritas en el capítulo 5, que debido a la falta de espacio no han podido incluirse en el cuerpo del TFG. Se corresponden con las figuras G.1 a la G.8.

G.2. Resultados

En esta sección se muestran algunos de los resultados gráficos obtenidos mediante el análisis de expresión diferencial.

G.2.1. Bowtie

Se muestran los diagramas de barras y las tablas con los resultados de Bowtie y las gráficas de las calidades de las lecturas de los experimentos. Se corresponden con las figuras G.9, G.10 y G.11.

G.2.2. Contador

Se muestra el diagrama de barras en el que se representa el conteo de genes. Estas son las figuras G.12 y G.13.

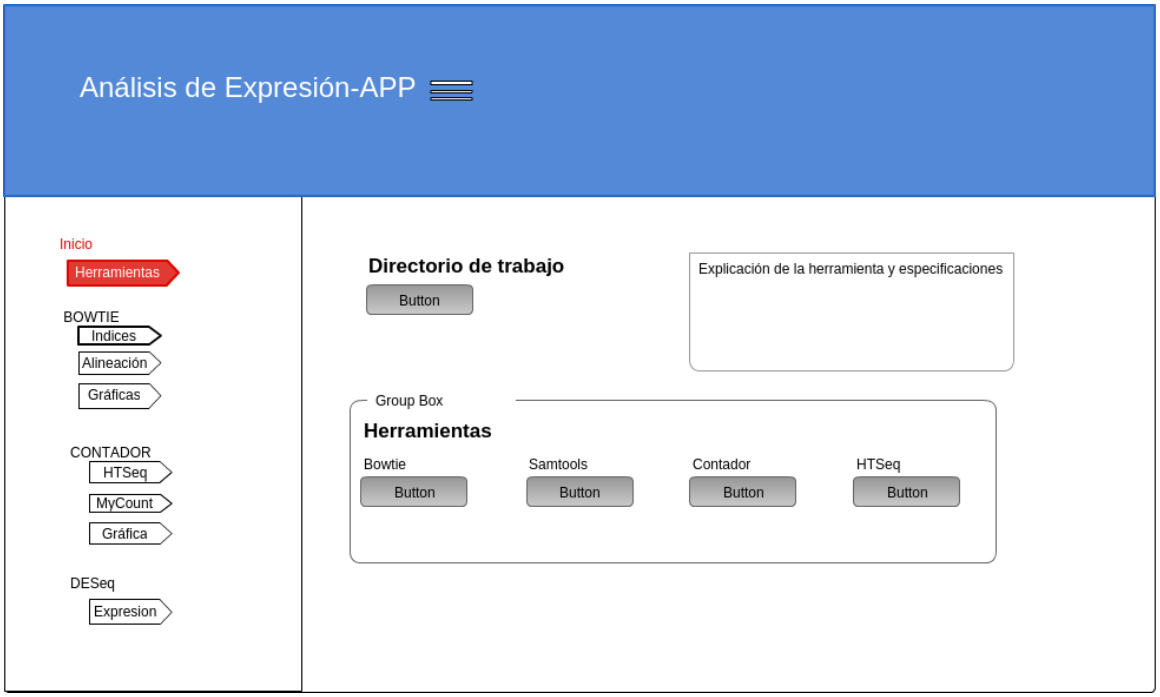


Figura G.1: Maqueta 1: Inicio: Determina el directorio de trabajo y las herramientas

G.2.3. DESeq

Resultados del análisis de expresión diferencial tras haber aplicado las funciones de la librería DESeq. Estos resultados se muestran en las figuras G.14, G.15 y G.16.

Figura G.2: Maqueta 3: Creación de índices

Figura G.3: Maqueta 4: Alineación

Figura G.4: Maqueta 5: Graficas de Bowtie

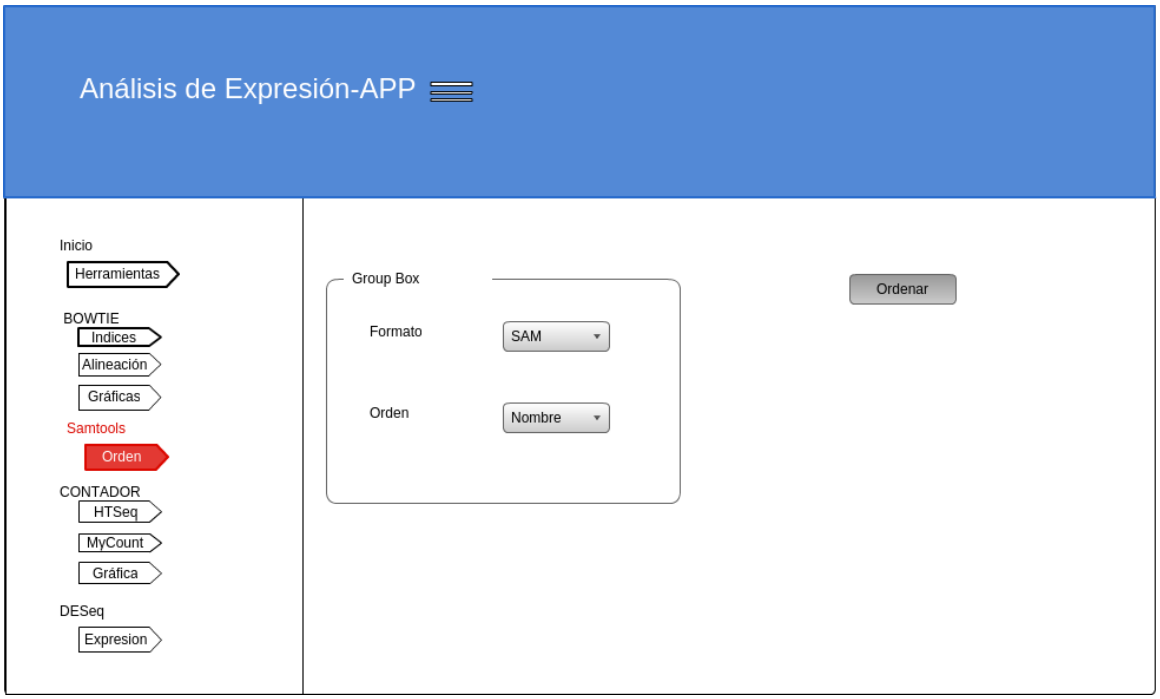


Figura G.5: Maqueta 6: Samtools

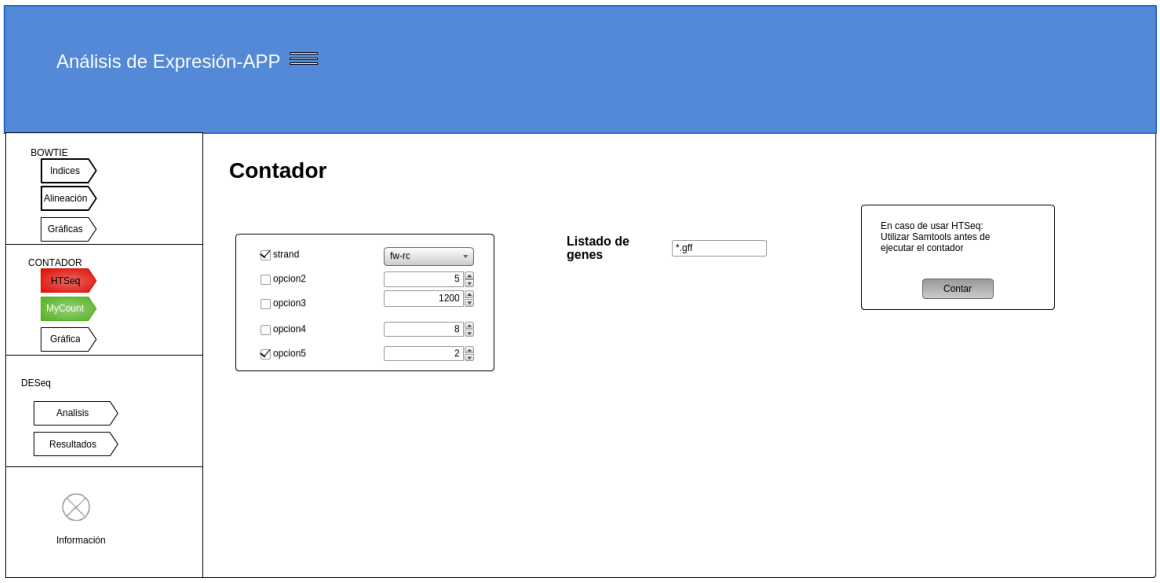


Figura G.6: Maquetas 8 y 9: Contadores

Figura G.7: Maqueta 10: Gráficas del conteo

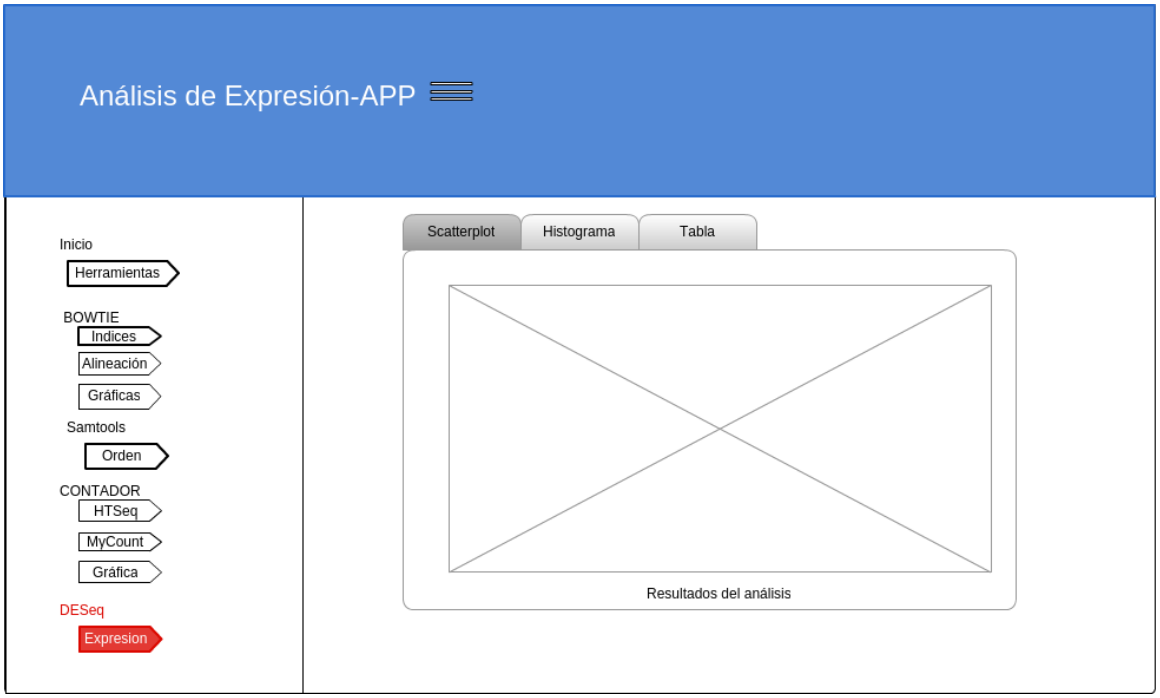


Figura G.8: Maqueta 11: Resultados del análisis de expresión diferencial

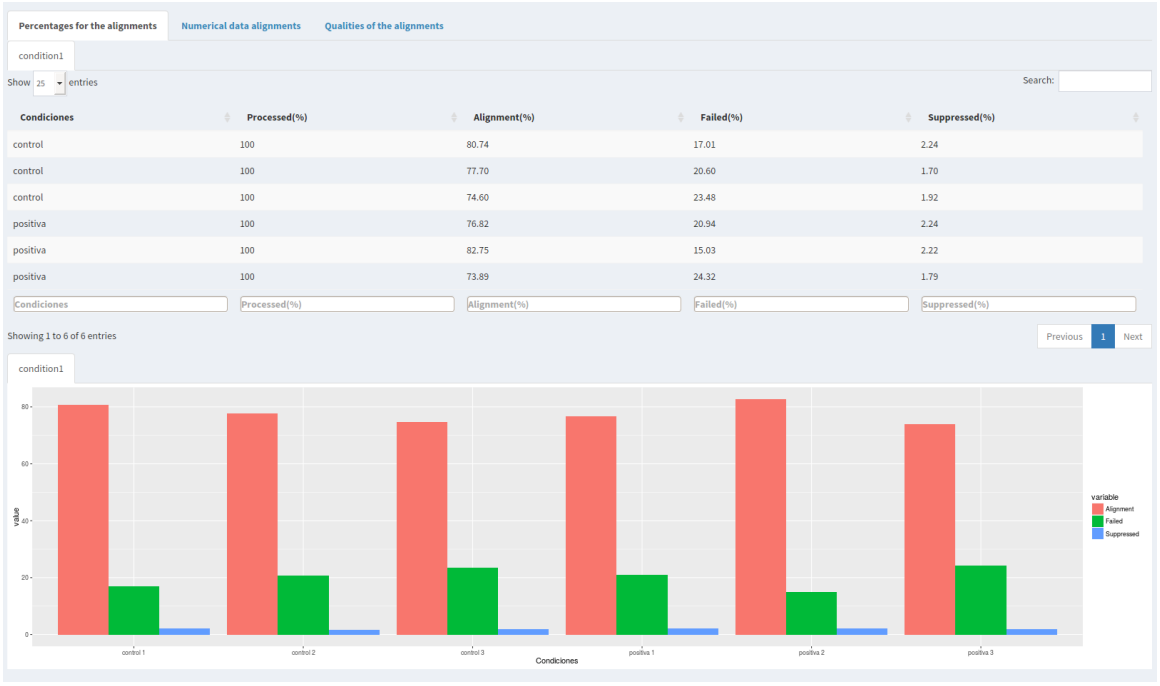


Figura G.9: Resultados de Bowtie en porcentajes.

Figura G.10: Resultados numéricos de Bowtie.

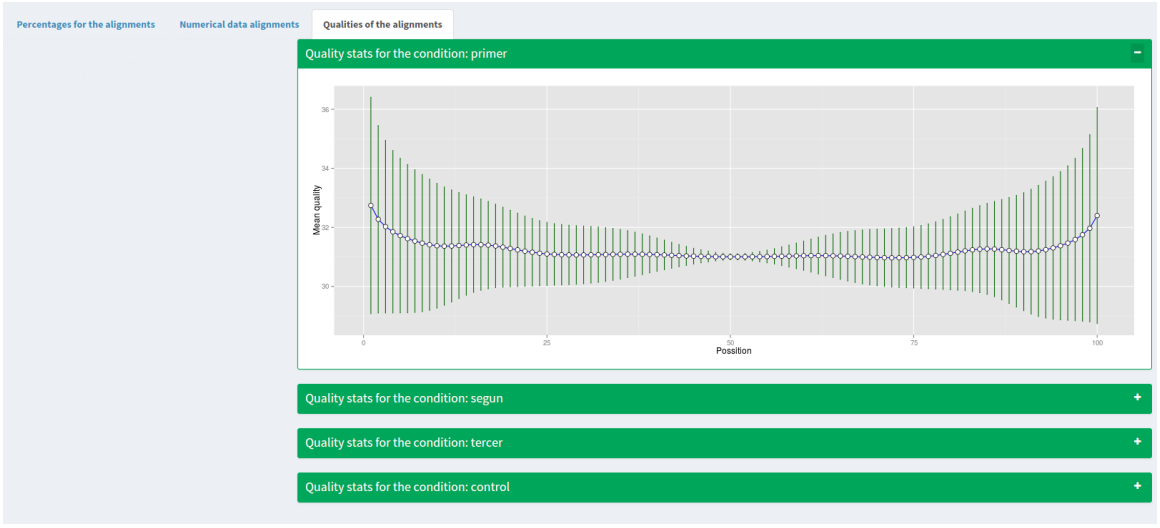


Figura G.11: Quality scores.

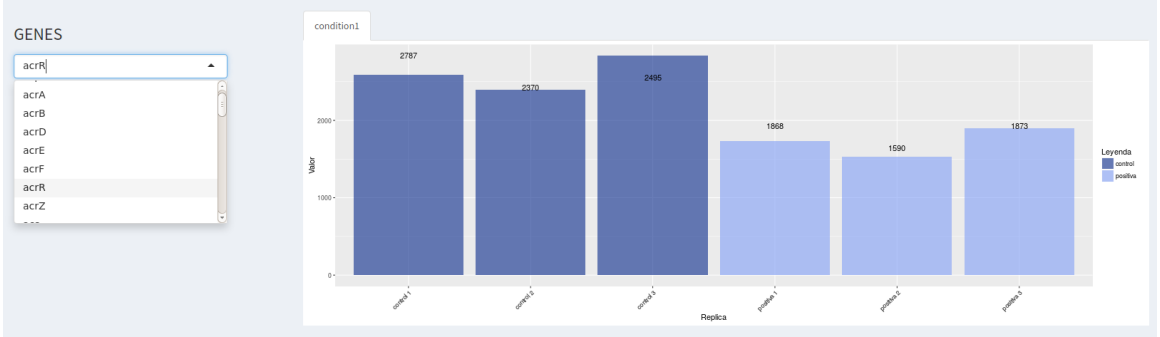


Figura G.12: Gen seleccionado acrR, y su conteo.

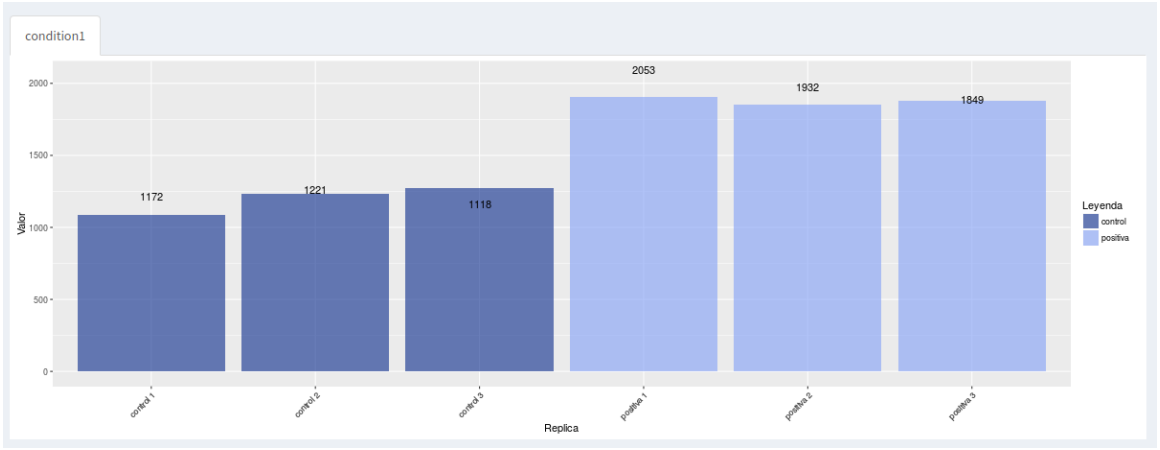


Figura G.13: Ejemplo de gen diferencialmente expresado

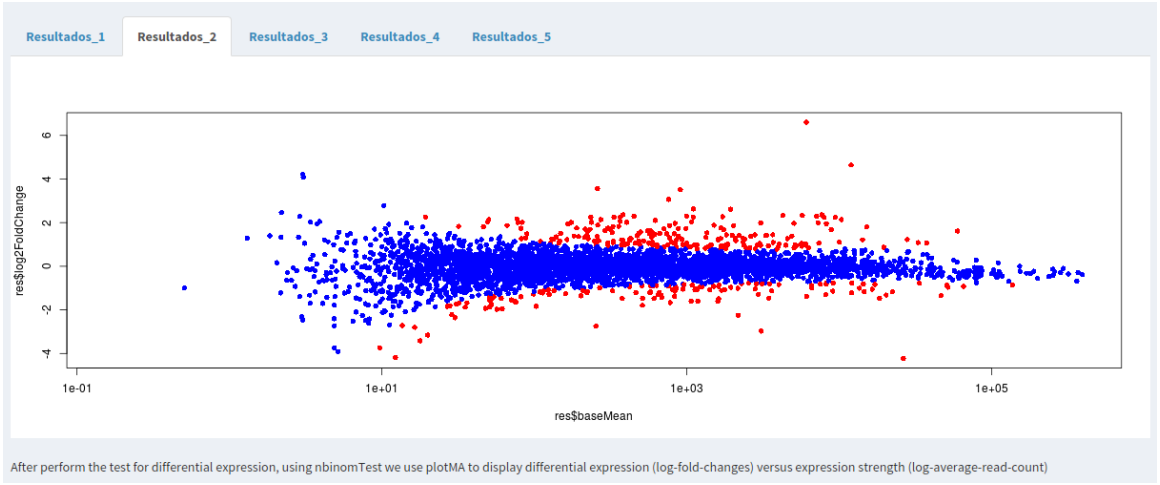


Figura G.14: Scatterplot de la media del conteo frente al fold-change en logaritmo

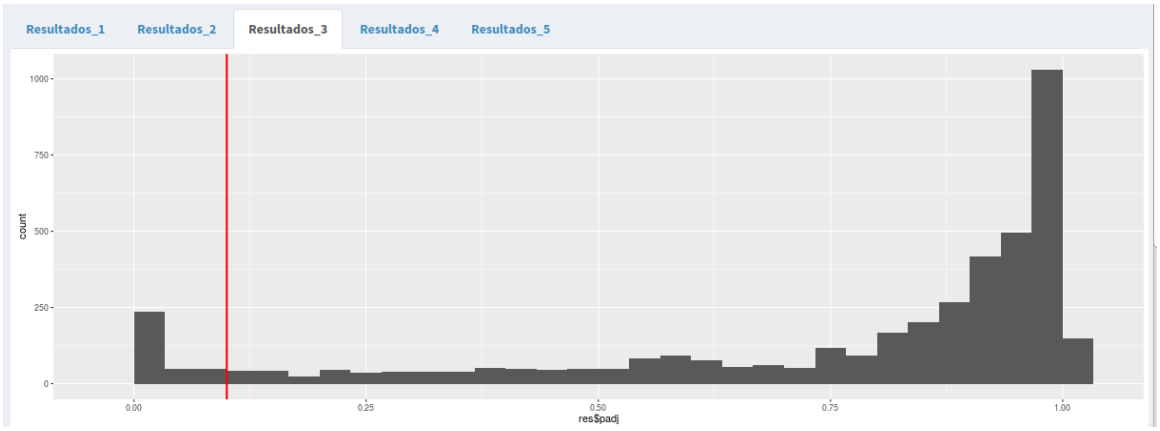


Figura G.15: Histograma de los p-valores ajustados, con separación por el FDR especificado

Resultados_1 Resultados_2 Resultados_3 Resultados_4 Resultados_5

Show 25 entries Search:

id	baseMean	baseMeanA	baseMeanB	foldChange	log2FoldChange	pval	padj
sgrT	6070.9227	123.32535	12018.52000	97.45376660	6.606646	8.094024e-74	3.423772e-70
setA	11943.1646	919.46425	22966.86487	24.97852940	4.642617	2.947368e-45	6.233682e-42
ptsG	26350.8185	50039.61613	2662.02083	0.05319827	-4.232477	5.981172e-40	8.433453e-37
leuD	907.7621	146.46856	1669.05574	11.39531766	3.510369	5.350107e-25	5.657738e-22
ibpA	3077.1742	5459.46819	694.88029	0.12727985	-2.973924	2.328751e-22	1.970124e-19
poxB	758.3320	161.96140	1354.70270	8.36435543	3.064254	4.777829e-19	3.368370e-16
ycaC	259.2987	40.51486	478.08260	11.80018026	3.560737	2.085201e-18	1.260057e-15
osmY	1935.0056	542.65070	3327.36055	6.13168015	2.616282	2.522527e-17	1.333786e-14
lldD	1103.4174	307.60943	1899.22539	6.17414549	2.626239	3.543088e-16	1.665251e-13
yjiY	7684.2922	2505.96458	12862.61987	5.13280194	2.359747	3.193096e-15	1.350680e-12
sucC	5721.9685	1896.97891	9546.95804	5.03271702	2.331337	5.178825e-15	1.991493e-12
gatC	4259.2438	1413.23913	7105.24846	5.02763355	2.329879	1.450925e-14	5.114511e-12
sucD	7111.6081	2411.14953	11812.06665	4.89893577	2.292468	1.705556e-14	5.549618e-12
sucB	9408.8680	3273.20616	15544.52977	4.74902252	2.247631	4.166377e-14	1.258841e-11
sucA	7958.1559	2778.12924	13138.18258	4.72914736	2.241580	7.308846e-14	2.061095e-11
ydjN	2172.6102	3587.66636	757.55398	0.21115508	-2.243625	1.236498e-13	3.268992e-11
gatD	1715.4368	588.46625	2842.40737	4.83019608	2.272082	3.151231e-13	7.841004e-11
deoC	10324.7837	3814.23144	16835.33588	4.41382127	2.142028	7.170761e-13	1.685129e-10

Figura G.16: Tabla de resultados del análisis DESeq

H

Cursos estudiados para el TFG

Para llevar a cabo este Trabajo de Fin de Grado ha sido necesario realizar un estudio de las bases de la bioinformática, de la genómica y de las herramientas utilizadas en el campo de estudio. Para ello se han cursado varios cursos ofrecidos por la *Johns Hopkins University* a través de la plataforma *Coursera* [43]. Los cursos del Programa Especializado en Ciencia de Datos Genómicos son los siguientes:

- Introduction to Genomic Technologies [44].
- Genomic Data Science with Galaxy [45].
- Python for Genomic Data Science [46].
- Command Line Tools for Genomic Data Science [47].
- Statistics for Genomic Data Science [48].